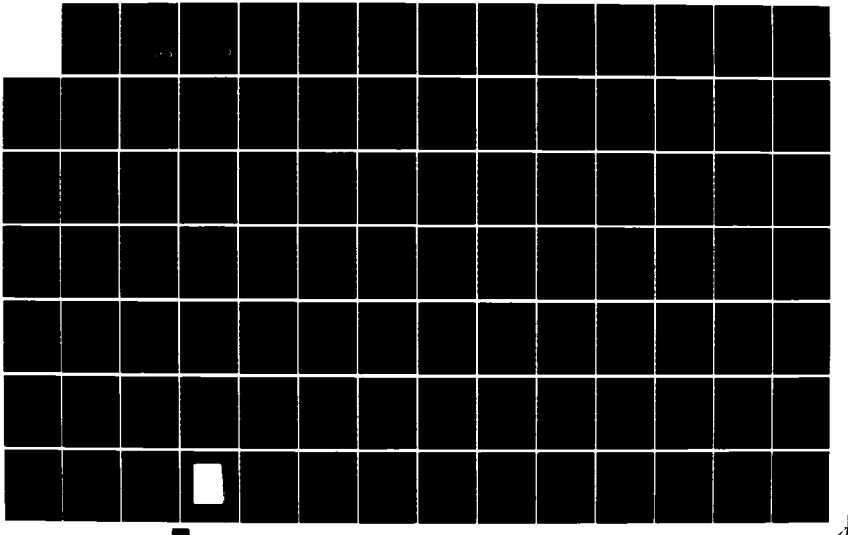
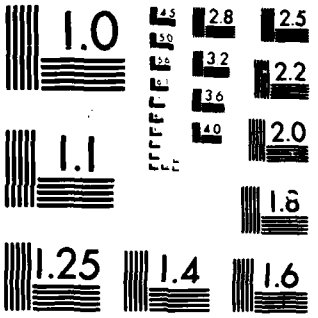
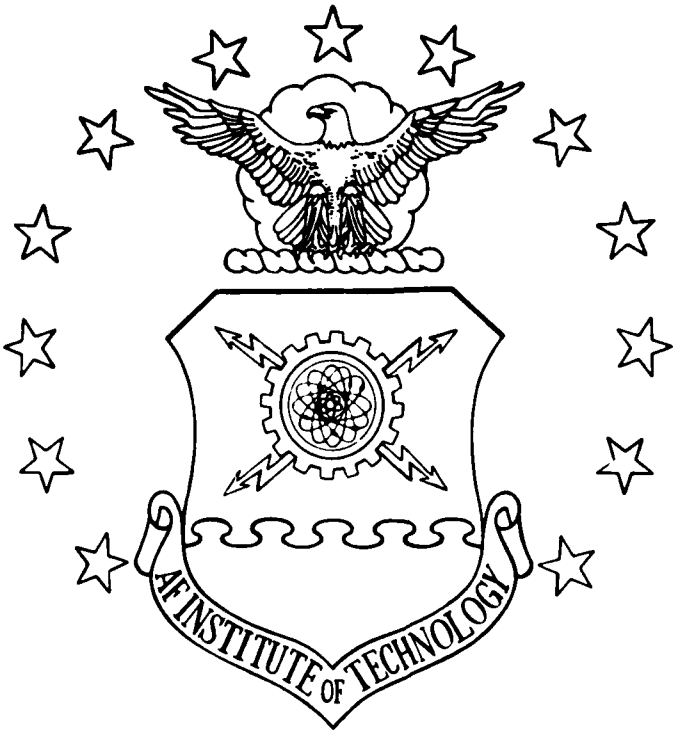
AD-A163 995 DESIGN AND IMPLEMENTATION OF HIGH PERFORMANCE CONTENT-ADDRESSABLE MEMORIES(U) AIR FORCE INST OF TECH WRIGHT-PATTERSON AFB OH SCHOOL OF ENGINEERING 1/2

UNCLASSIFIED W H SWINN DEC 85 AFIT/GE/ENG/85D-39 F/G 9/5 NL







DESIGN AND IMPLEMENTATION OF HIGH PERFORMANCE CONTENT-ADDRESSABLE MEMORIES

THESIS

Wook H. Shinn
Captain, USAF

AFIT-GE/ENG/85D-39

DEPARTMENT OF THE AIR FORCE
AIR UNIVERSITY
AIR FORCE INSTITUTE OF TECHNOLOGY

Wright-Patterson Air Force Base, Ohio

AFIT-GE/ENG/85D-39

DESIGN AND IMPLEMENTATION OF HIGH PERFORMANCE CONTENT-ADDRESSABLE MEMORIES

THESIS

Wook H. Shinn
Captain, USAF

AFIT-GE/ENG/85D-39

AFIT-GE/ENG/85D-39

DESIGN AND IMPLEMENTATION OF HIGH PERFORMANCE CONTENT-ADDRESSABLE MEMORIES

THESIS

Presented to the Faculty of the School of Engineering
of the Air Force Institute of Technology
Air University
In Partial Fulfillment of the
Requirements for the Degree of
Master of Science in Electrical Engineering

by
Wook H. Shinn, B.S.
Captain USAF
Graduate Electrical Engineering

December 1985

Preface

This thesis looks into strengthening the relationship between artificial intelligence (AI) and very-large-scale integration (VLSI) semiconductor circuits. Originally, ways to use AI to assist VLSI design were studied. However, in the end, the reverse situation was explored - how can VLSI assist AI by the design of high performance content-addressable memory.

Since early undergraduate studies, I have had a strong interest in computer engineering and VLSI design. My AFIT tour has given me the opportunity to further my education in these fields. Selection of this thesis topic as the culmination of my curriculum proved to be a tremendous and successful learning experience.

I offer special thanks to Dr. Richard Linderman, my thesis advisor, for urging me to take on this thesis project and for his invaluable help, guidance, and draft reviews. I also wish to credit him with the design of some of the CAM subcells. I would also like to thank Dr. Harold Carter for his help in reviewing the draft.

Special thanks goes to my lovely wife, Inhee, for her assistance in drawing figures and editing drafts of this thesis, and most of all for the encouragement she gave me.

Wook H. Shinn

Contents

List of Figures

List of Tables

AFIT-GE/ENG/85D-39

Abstract

The content-addressable memories (CAMs) have major applications in developing areas of artificial intelligence and image processing. This work involves researching, designing, and implementing a high performance, high density CAM. The design, fabrication, and test of this all-parallel AFIT CAM is discussed. The AFIT CAM stores 2K bits of information and fits in 64 pin dual-in-line package. It was fabricated using a 3u CMOS technology with 2-layer metals.

The AFIT CAM supports four types of comparisons directly on the chip. These are equal, not-equal, greater-than-or-equal, and less-than. In addition, it has DUMP and Parallel Write operations to aid artificial intelligence and image processing applications.

The AFIT CAM has a total of 78,000 devices and achieves a density of 1,073 devices per mm^2. The memory access time for reading and writing are only 22 ns and 20 ns respectively. The CAM chip computes 800 million magnitude comparisons or 3.2 billion equality comparisons per second.

The cif file has been resubmited to the manufacturer for fabrication due to a short circuit which was caused by over bloated second metal lines. A second version of the AFIT CAM cell has been redesigned and nets a 25% reduction in size.

DESIGN AND IMPLEMENTATION OF HIGH PERFORMANCE CONTENT-ADDRESSABLE MEMORIES

1. Introduction

1.1. Background

The random access memory is only a limited means to solve the access requirements of many problems, since it primarily involves storage space with a fixed, compact, unique code label for each object in the program. It would be much easier for problem oriented languages, to use "names" which would not directly correspond to the expensive address space. A Hash algorithm is one of the methods used to compress the name to an address. Another well known method is the insertion of this name into an ordered list and then performing binary search for retreival. To implement this in hardware, each memory cell must have sufficient logic to determine whether or not it holds data that matches with some criterion broadcast from the central unit of computer.

In computer technology, the name, associative memory has generally been adopted. The other common used name for this type of devices is content-addressable memory (CAM), this name used here because the device under considerations implements only a limited set of operations dealing with in associative recall.

Designers and users of computer systems have long been aware of the fact that some kind of content-addressable or

"associative" functions would allow a more effective way to search data in the memory. However, the hardware content-addressable memories have primarily been used in special roles such as small buffer memories and control units.

Now, This situation seems to be changing because of the development of new technologies especially VLSI semiconductor circuits. This VLSI technology allows the active searching functions to be transfered into the memory unit. The price of this more complex memory has been greatly reduced to allow much broader application of these principles. This certainly will have a significant impact on the future computer architectures, especially those solving problems in artificial intelligence.

1.2. Problem Statement

The object of this thesis is researching, designing, and implementing an high density, high performance content-addressable memory to serve as a hardware building block for future architecture work in artificial intelligence, image processing, and database management systems.

1.3. Scope

In addition to this thesis, the VLSI CAM research project encompasses other area, Content-Addressable Memory Manager - Design and Implementation, which is addressed by Captain Mark Rowe.

1.4. Overview of Thesis Contents

Chapter 2 contains a detailed problem analysis and some general information on the design approach that is applicable in Chapter 3. It begins with a discussion of the basic concept of content addressability and two basic implementations of content addressing. The need and application of hardware CAM is presented to motivate the topic and emphasize the importance of the CAM. Next, the basic organization of all-parallel CAM is discussed. This will give the reader a general overview and basic components of the CAM. The explanation and logic of the each components is presented. In order to determine the advantages and disadvantages of other types of CAM, the word-parallel, bit serial CAM organization is briefly discussed. Last, the use of all-parallel CAM and word-parallel, bit-serial CAM is discussed by giving examples.

Chapter 3 discusses the actual implementation of the AFIT CAM. It begins with the organization of the AFIT CAM. The basic components are same as the all-parallel CAM discussed in Chapter 2. However, AFIT CAM has more components to implement added special features. The CAM chip can be divided into three parts according to their functions: CAM cell, RAM circuitry, CAM circuitry. Rest of this chapter discusses these parts.

Chapter 4 discusses the operations of the AFIT CAM. The reading and writing operations are explained by using timing diagrams. The searching operation are also explained in this

chapter by using search algorithms.

The results of this thesis efforts are given in Chapter 5 along with the conclusions and recommendations.

2. Detailed Analysis of the Problem

Many data and signal processing applications such as radar tracking, image processing, computer vision, and artificial intelligence require fast storage, searching and retrieval of data from memory (2:375). Traditionally, to search a memory one stores all items where they can be addressed in sequence. The search procedure is a strategy for choosing a sequence of memory addresses, reading the content of each address, and comparing the information read with the item being sought for until a match occurs. The number of times memory is accessed depends on the location of the item being sought for and the efficiency of the search algorithm. Many search algorithms have been developed to minimize the number of memory accesses while searching for an item in a random or sequential access memory. In general, sequentially searching through an ordered list in memory requires at least time proportional to the logarithm of the number of items in the list.

The time required to find an item stored in memory can be reduced considerably if the stored data can be identified for access by the content of the data itself rather than by the data's address. A memory accessed by its content is called an associative memory or content-addressable memory (CAM). We can define a content addressable parallel processor (CAPP) to be a CAM with the added ability to write in parallel to all words in memory, part of the words, or

even just a single bit in a subset of the words (3:3). This is called the multi-write capability and distinguishes CAM from CAPP.

2.1. Basic Concepts of Content Addressability

Accessing data on the basis of content always requires some comparison of an external search argument with part or all of the information stored in the memory cells. A "genuine" CAM performs all of its comparisons in parallel. In principle, whether this is accomplished by software, parallel electronic circuit or mechanical scanning is immaterial.

In addition, comparison by an equality match between the search argument and the stored item is not the only useful mode. If the stored data represent numerical values, the purpose of content-addressable search may be to locate all cells that satisfy certain magnitude relations with respect to the search arguments. Examples of this are being greater than or less than a given limit, or between two specified limits. Content-addressable searching is sometimes performed without reference to an external search argument. Locating the maximum or minimum in a set of stored numbers uses this aspect of the searching process. Finally, searching on the basis of the best match of the search argument to the various stored items, in the sense of some metric, may be necessary. This resembles the basic task of pattern recognition (1:5).

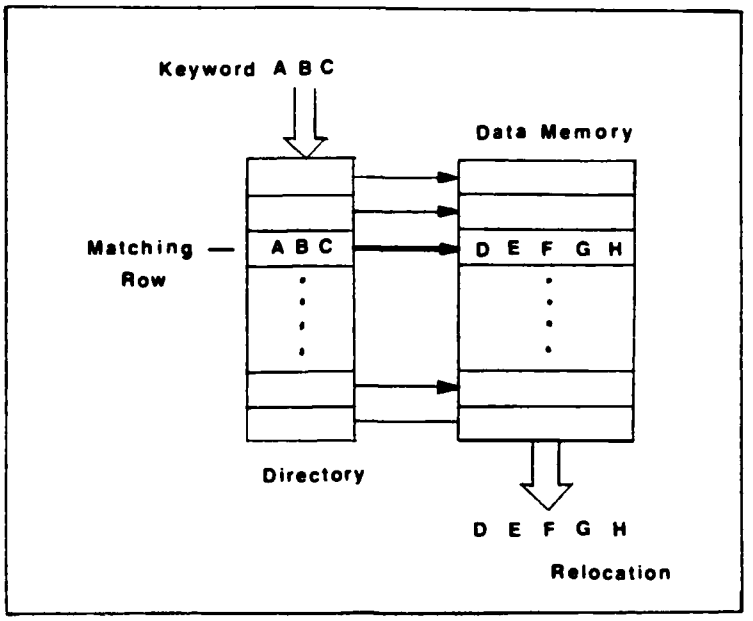


Figure 2.1. Catalog Memory

To illustrate, consider the catalog memory system shown in Figure 2.1. It consists of two parts, the directory and the data memory. This system only compares the search argument, here called keyword with all words stored. It is assumed for simplicity that all of the stored words are different so that at most one match is expected. Every word location in the directory contains a register for each stored word, as well as a special combinational logic circuit which compares the keyword, broadcast in parallel to all locations, with the data memory contents.

Each location has an output from which a response is obtained if the keyword and the stored word agree. One response, as shown in Figure 2.1 by a bold arrow, will be obtained. The data memory is usually a linear select

random-access memory for which the output lines of the directory serve as address lines, eliminating the need for an address decoder. To put it another way, the directory assumes the role of the address decoder, with the operation of the latter being dependent on the words stored within. The whole system can be regarded as a special logic circuit with an input-output signal delay access time of the CAM that is very short.

Two auxiliary features which are commonplace in CAMs are a provision to mask off parts of the search argument, and an organization which allows the sequential readout of several responding items (1:6).

2.2. Two Implementations of Content Addressing

The two most common implementations of content addressing are (1) data-dependent memory mapping implemented by programming techniques (software), and (2) special hardware constructs for the storage and retrieval of data items. It is interesting that both of these concepts were invented at about the same time in 1955. With the advent of the first commercial computer systems there existed a strong need for content addressing. While over thirty years have passed since the introduction of these methods, essentially no new arguments have been presented in favor of one of the other. Therefore, both approaches are reviewed here.

2.2.1. Software Implementation: Hash Coding

Before the introduction of the first automatic

programming languages, such as FORTRAN, a need existed for assemblers which could refer to the computational variables by their symbolic names. Translation of a name into the address of a storage location at which the actual variable resides necessitates maintenance of a table. Frequent table look-ups are then required. Instead of scanning all the stored entries every time (i.e., performing a linear search), much quicker methods were invented.

Since every name has a binary representation in the memory, it thereby also has a numerical value. In principle, if the address space of the memory system were unlimited, the name could be stored at an address which equal its numeric value, and any data associated with the variable could be retrieved in a single access to the memory. In practice, the available memory is much smaller than the range of numbers spanned by all permissible names. Obviously, some sort of comparison on the name's space of numerical values is necessary.

Assume that the names are represented by a 26-letter alphabet, and only the first two letters are required to determine the numerical value; there are $26^2 = 676$ different pairs of letters. A memory area of this size would be reserved for the table.

Now assume that random names are used. Table 2.1 is a set of randomly chosen first names of persons used as

Table 2.1. Randomly chosen names, and their numerical values computed from the first two letters (1:7)

Sample No.	Name	Value (address)	Associated data
1	ISABEL	226	D(1)
2	SOPHIE	482	D(2)
3	HOWARD	196	D(3)
4	JULES	254	D(4)
5	EDWARD	107	D(5)
6	JOHN	248	D(6)
7	WILLIAM	580	D(7)
8	GEORGE	160	D(8)
9	JESSE	238	D(9)
10	NIGEL	346	D(10)
..	.....	...	.....
14	HAROLD	182	D(14)
15	WALTER	572	D(15)
16	HARRY	182	D(16)

indentifiers. Their numerical values are given in the table's third column. These values are calculated in the following way: A=0, B=1, ..., Z=25. A pair of letters is regarded as an base 26 integer. The numerical value of the first letter is multiplied by the base, 26, and added to the value of the second letter to arrive at the value of the letter pair, e.g., IS = 8 x 26 + 18 = 226. The address in the table which is defined by the numerical value of the name, is the name's calculated address. At sample 16 a name which has the same two beginning letters as sample 14 was detected. A conflict or collision has occurred. Since both names cannot be stored in the same location, a reserve location, obtained from the calculated address, must be found. One way to solve this problem is the use of the next empty location, following

the calculated address in this case location 183. If a search for an empty location is made cyclically over the table, such a location can be found sooner or later as long as the table is not completely full. If the table is not more than half full, and the names are chosen randomly, there is a high probability of finding an empty location within a few locations of the originally calculated address.

To resolve whether an entry is stored at the calculated address or at one of its reserve locations, the name of its unique identifier must be stored together with the data. Assume that the name itself is used. The correct location is found when the stored name agrees with the name used as the search argument. Table 2.2 exemplifies the contents of part of the memory, corresponding to the example defined in Table 2.1. A search for the data associated with HARRY is

Table 2.2. Partial contents of the memory area used to store the data corresponding to Table 2.1

Address	Contents of the location	
	Name	Data
...	...	...
160	GEORGE	D(8)
...	...	...
182	HAROLD	D(14)
183	HARRY	D(16)
...	...	...
196	HOWARD	D(3)
...	...	...

easily performed. The calculated address of HARRY is 182.

Since such an identifier is not found there, a search from the next location reveals that HARRY was stored at address 183. The associated data, D(16), is then found.

Obviously, the average number of trials to find a stored item from the neighborhood of its calculated address depends on the loading of the table and its size. Moreover, the number of trials is smallest if the items are scattered uniformly in the table. Instead of using sampled letters as in the above introductory example, a better method is to compute some function of the whole name which spreads its calculated address deterministically over the available memory area. Hash coding is such a method, and the address mapping is defined by the hashing function.

2.2.2. Hardware Implementation: The CAM

Next consider parallel searching using a special content-addressable memory (CAM), corresponding to the directory depicted in Figure 2.1. Assume that at the word locations binary patterns with bit elements "0" or "1" are stored. The logic for two word locations is depicted in Figure 2.2. The search argument is broadcast via a set of vertical bit lines each carrying a bit value to its respective position for all word locations. Built-in logic circuitry, consisting of a logical equivalence gate at every bit position, and an n-input AND gate (for n bit comparisons) for each word, is shown in Figure 2.2. The logic circuit gives a "1" response if and only if the search argument

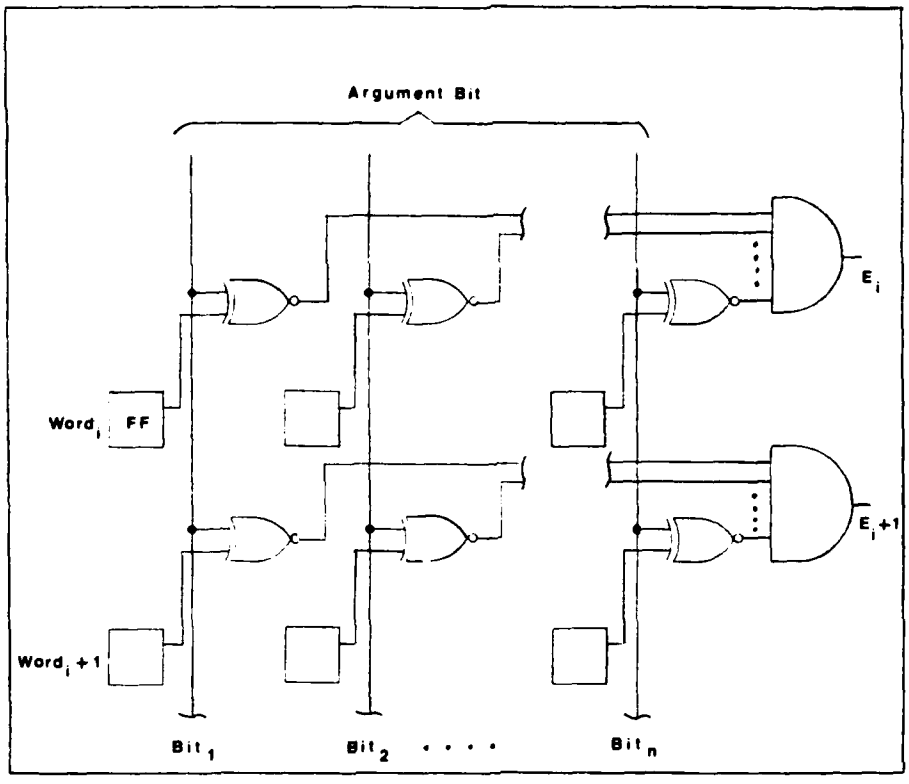


Figure 2.2. Comparison Circuit for All-Parallel CAM

agrees with the stored binary pattern. This circuitry is repeated at every word location of the directory in Figure 2.1.

2.3. Needs and Applications for Hardware CAMs

The conventional Von Neumann computer suffers from a serious fundamental handicap stemming from the principles of addressing its memory. Each operand must be read and written serially, which consumes a great deal of time. Many large problems solved faster if operand could be accessed in parallel. With increasing memory capacity, the address length

increases which means that more space must be reserved in each memory location which contains an address reference. For this reason, instead of addressing the whole memory system directly, it is customary to divide the memory space of a large computer into several smaller banks. The banks are accessed individually, using indexing of the banks in machine instructions, as well as relative and indirect addressing for the reduction of the address range handled within the program. For the determination of the absolute addresses within the memory, several auxiliary arithmetic operations must be performed for each memory reference.

In order to operate on many memory locations simultaneously, and to simplify searching for operands, it would be desirable to base the computations on content-addressable memories. During the past 20 years several CAM developments have been attempted. Content-addressable memories would be especially advantageous from the point of view of high-level programming languages which refer to their operands and procedures with symbolic names and other identifiers. The computer architectures would be quite different from those currently applied if highly parallel computations, distributed all over the data files were simply and directly implementable in hardware. Unfortunately, large content-addressable memories have not been realized for systems requiring direct access to data on the basis of their symbolic representation, i.e., by names or alphanumeric

identifiers. It seems that software methods such as hash coding are still prevalent. And unfortunately, no parallel processing is involved in software content addressing.

Nonetheless, hardware content-addressable memories have been used as special parts in computing systems. In certain organizational solutions CAMs can effectively perform fast buffering, checking, and bookkeeping operations needed to move and rearrange data. In this way, although the memory system as a whole is not content-addressable, the CAM devices can effectively contribute to the speed of operations by making the operands more directly available to the processing circuits. Two such applications are the virtual memory and dynamic memory allocation circuits.

The ways in which data are specified in a content-addressable search, differ from one application to another. As an example, for symbolic computations it is necessary to locate all entries which exactly match a given search argument, for instance, an identifier in its special portion. In the retrieval of numerical data from large files, there frequently is a need to locate all entries which satisfy certain magnitude relations. These entries have one or more of their attributes above, below, or between specified limits. Much effort has thus been devoted to the development of special CAM devices with an extremely high storage capacity. Progress has been slow, but smaller size CAMs can be helpful. Although the available content-addressable memory capacity is too small to hold all files, it can be

used in a buffering mode for the execution of searches on large sections of data. In this way, even though many batches of data must be processed sequentially, the overall searching time can be reduced several orders of magnitude.

CAMs have already had a significant impact on certain parallel computer architectures. There exist a few types of content-addressable parallel processors which have been developed for special applications, e.g., air traffic control, numerical transformation, and digital signal filtering in which many computations are carried out.

Other areas of application of the CAMs and their derivatives are in the programming of logic operations in the central control circuitry of computers and also in other equipment where numerous Boolean conditions have to be evaluated or table look-ups performed.

2.4. Basic Organization of the All-Parallel CAM

The block scheme of an all-parallel CAM system is shown in Figure 2.3. The basic elements are the CAM array, the search argument register, the mask register, match logic circuit, the match store, the priority resolver, encoder and some auxiliary circuits needed for interconnection and control (1:142).

The mask bits are stored in the mask register which, together with the search argument, form the comparison signals fed into the CAM array.

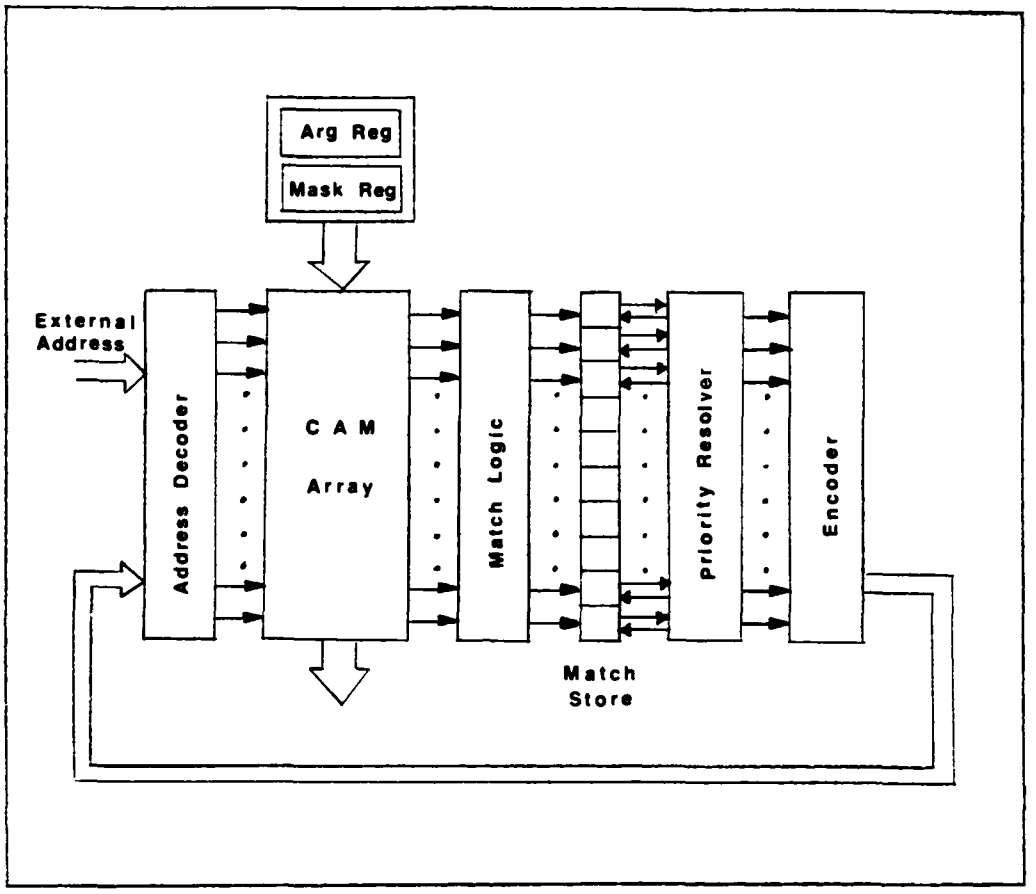


Figure 2.3 All-Parallel CAM Organization

The CAM array consists of the memory cells to store data, a magnitude detector and an exact match indicator. The outputs of the CAM array are BORROW (Bout) and EQUAL (E) signals that are generated by the magnitude detector and the exact match indicator. These two outputs for every word are fed into the match logic circuit. There the match determined depends on the external control inputs which relect the type of the match. The match signals are then stored in the match store element which is a bank of flip-flops. If more than

a single match occurs, we need some method to output them one at a time. The priority resolver solves this problem.

The output lines of the priority resolver could be connected directly to the address lines of the CAM. This allows the complete contents of all the matching words to be read one item at a time. For normal addressed reading and writing there is a built-in address decoder. For this reason the output from the priority resolver must be encoded, as shown in Figure 2.3.

The provisions for masking the search argument are useful in several applications. The most important of these may be the search for selected attributes which are parts of the search argument. The words in the content-addressable memory may be composed of several fields describing different attributes, and a subset of these can be selected by umasking the corresponding search argument.

Another important application is finding an empty location for new data to be written into memory. Empty places in memory must be found automatically. To allow this, there is an additional vacancy indicator bit in each word which is comparable to a data bit. A "1" in this bit indicates the presence of information. After deletion of the entry, this bit is out reset to "0". The vacant places can be found by masking all other bits except the vacancy indicator bit, and performing an equality search. The vacant location into which a word is written is then determined using the priority resolver and the encoder.

A third possible use of masking is for the magnitude search on numerical entries. This use will be discussed in section 2.7.

2.5. Logic Principles of All-Parallel CAM

The physical as well as mathematical principle necessary for developing the logic circuitry of all-parallel CAMs will be discussed in the following sections. Discriptive Boolean expressions as well as device characteristics are discussed to provide a well rounded explanation of the logical operation of the all-parallel CAM.

2.5.1. Data Storage

The static CMOS memory cell can be used in the CAM array to store a data bit. The data can be read and written into the memory cell. A typical static CMOS memory cell is shown in the Figure 2.4. Although this is a static cell, in that data retention is permanent (as long as V_{dd} is present).

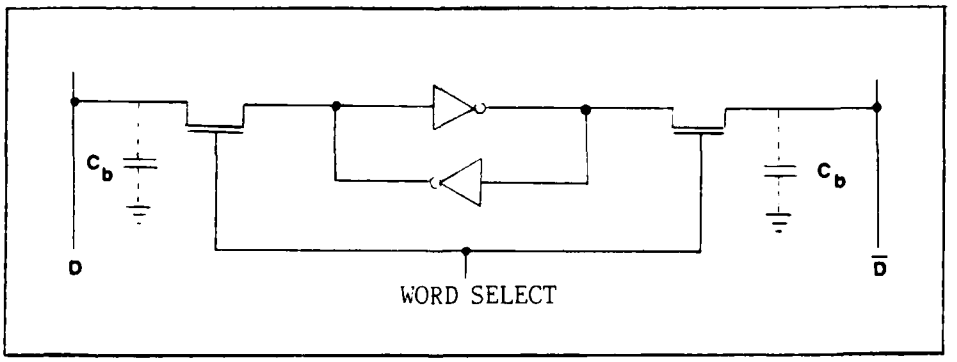


Figure 2.4 Static RAM cell

When the two inverters are connected as in Figure 2.4, the characteristics of two types of devices are effectively multiplied together. And the transfer charateristics of the two-in-series becomes extremely nonlinear. The cross connection provides large noise margins, and well-defined output levels of ground and supply voltage. Since each leg of the flip-flop always contains one conducting and one nonconducting trasistors, there is always a relatively low impedance path from the output terminal to either the power supply or ground, depending upon the state of the inverter. The low impedance minimizes the capacitive noise pick up. The switching response of these CMOS logic circuit is determined solely by the capacitance loading at the input and output and the amount of current available to charge this capacitance (9:265). Note that in the Figure 2.4 the bitline capacitance, C_b, is illustrated. This is the result of parallel stacking of a number of cells on the bitlines.

The equivalent circuit of a RAM cell with peripherals is shown in Figure 2.5. To write the cell, DATA is placed on the D line and $\overline{DATA}$ is placed on the $\overline{D}$ line by openning the p-devices when $R/\overline{W}$ is zero. Then, the WORD SELECT is asserted.

During a read operation ($R/\overline{W}$ = 1 and PRE = 1) the both bitline are precharged to V_{dd}. When the pass gates are turned "on" the logic "0" side of the cell discharges the bitline capacitance through the pass gate and the pull-down transistor associated with the inverter in the "0" state.

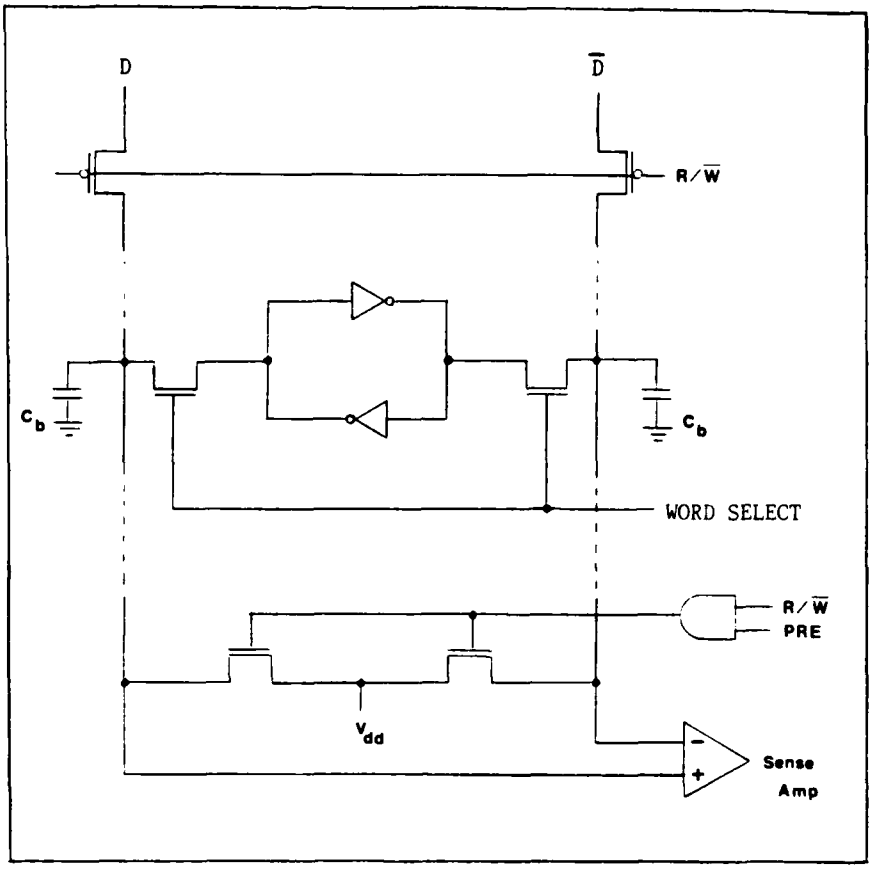


Figure 2.5 Equivalent Circuit of RAM Cell with Peripherals

This process is reasonably slow, hence the state of a memory cell is detected with a differential amplifier sensing the outputs of D and $\overline{D}$.

The sense amplifier can be realized by using devices of CMOS technology to imitate conventional bipolar differential amplifier configurations with complementary current mirror loads. Figure 2.6 shows a CMOS differential gain stage configuration with current mirror loads. The available voltage gain, assumming an open-circuit load, can be given as

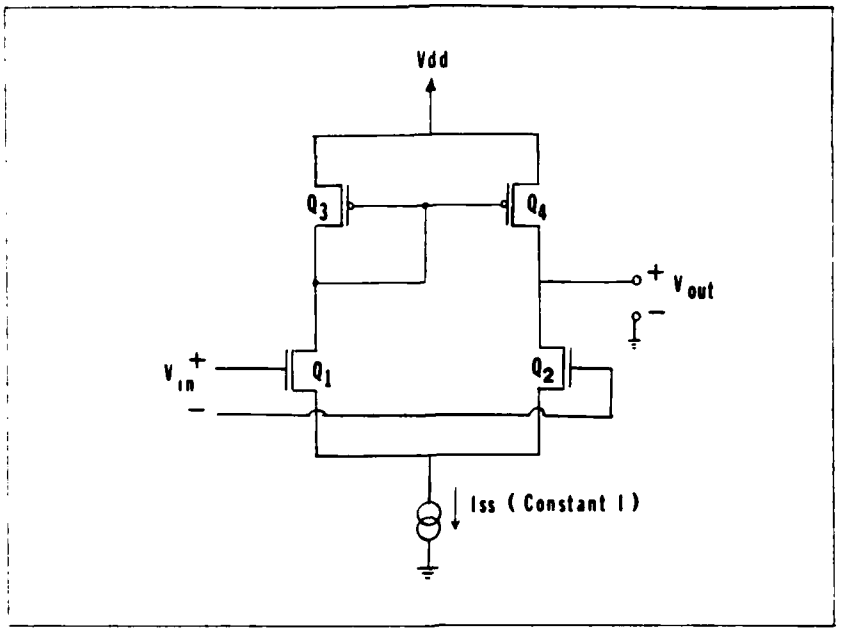


Figure 2.6 CMOS Differential Gain Stages with Current Mirror Load (5:294)

$$A_v = \frac{V_{out}}{V_{in}} = -G_m(r_{o2} // r_{o4}) \quad (5:294)$$

where,

Gm = transconductance

$$Gm = \alpha(\Delta I_D)/\alpha(\Delta V_i) = I_{ss}(\mu_s C_{ox})\left(\frac{W}{L}\right) \quad (5:279)$$

u_s = carrier mobility in the conducting channel
C_{ox} = capacitance per unit area of the gate region

W/L = ratio of width to length of the gate
r_{02}, r_{o4} = output resistance of Q2 and Q4.

With practical bias levels and device geometries, typical gains from such stages range from 100 to 1000. These values of gain can be increased to the range of 1000 to 10,000 by using a cascade configuration for Q1 and Q2, and by values of Gm in typical CMOS transistors (5:294).

The actual sense amplifier and the peripherals used in the memory circuit for the AFIT CAM are slightly more complicated than the circuit described in this section. In the AFIT CAM, a two stage CMOS differential amplifier is used. This circuit will be discuss at greater length in Chapter 3.

2.5.2. Equality Comparison

A basic operation of hardware content-addressable memory (CAM) structures is bit matching. If the Boolean values of two binary variables are denoted by A and B, then the Boolean function which has the value 1 (true) for logical equivalence (match) of A and B, and 0 (false) is:

$$(A \equiv B) = AB + \overline{A}\,\overline{B} \tag{2.1}$$

Alternatively, the mismatch of the values A and B is indicated by the EXCLUSIVE OR-function:

$$A \oplus B = A\overline{B} + \overline{A}B \tag{2.2}$$

In a masked search, only a subset of the bits of the search argument word is compared with respective bits of all memory words. (Masking in the CAM usually means masking out, or excluding certain bits from the match.) Those stored words which agree in the specified unmasked bit positions of the search argument are flagged. For this reason each bit position in the memory is equipped with XOR logic, together with an indication as to whether that bit is involved in the

comparison or not. Let the jth memory word be $A_j = (a_{j0}, .., a_{jn})$, with the a_{ji}s as its respective bits, and let boolean search argument be $B_j = (b_{j0}, .., b_{jn})$. If the mask word $C = (c_0, .., C_n)$ has the Boolean value "1" in all masked (disabled) bits and all other bits are "0", the masked match of the respective bits is defined by the function in the ith bit position:

$$m_{ji} = (a_{ji} \equiv b_{ji}) + c_i \ . \tag{2.3}$$

Note that when the ith bit is masked, m_{ji} is indentically "1" because c_i is equal to "1". The masked mismatch in the ith bit position is indicated by the Boolean expression:

$$m_{ji} = (a_{ji} \oplus b_{ji}) \cdot c_i \ . \tag{2.4}$$

The matching of the memory word A_j with the search argument B_j in all unmasked positions is defined by requiring a match condition at all bit locations.

$$m_j = \bigwedge_{i=0}^{n} m_{ji} \ . \tag{2.5}$$

2.5.3. Magnitude Comparison

Another important operation in content-addressable search is the identification of every memory word which has a numerical value greater or less than a given search argument. The between-limits comparisons, as well as the search for the extremes, can be executed by combining two or more of these operations in a memory system. For simplicity, all binary

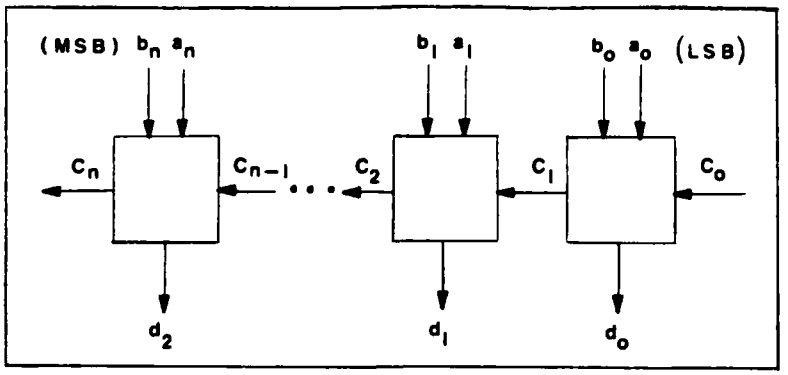


Figure 2.7 n-bit Subtractor

operations in a memory system. For simplicity, all binary words represent nonnegative integers in the binary number system.

The magnitude comparison may also be realized by an arithmetic operation using full subtractors. Let us assume that the general cell shown in Figure 2.7 is a 1-bit full subtractor. The magnitudes can be compared by subtracting B from A using the n-bit subtractor. The output C_n is then a Borrow signal. If there is a Borrow ($C_n = 1$) then word A is less than word B. When there is no Borrow ($C_n = 0$) then either A is greater than or equal to B. In order to determine whether or not A is equal to B, we must examine the difference bits d_0 through d_n. If d_0 through d_n are all zeros, we know A is equal to B. However, the logic circuit for generating the difference of two bits is twice as complicated as the XOR (or XNOR) gate which can perform the equality comparison. Therefore, we use only the Borrow signal in the full subtractor for a magnitude detector. A cell of

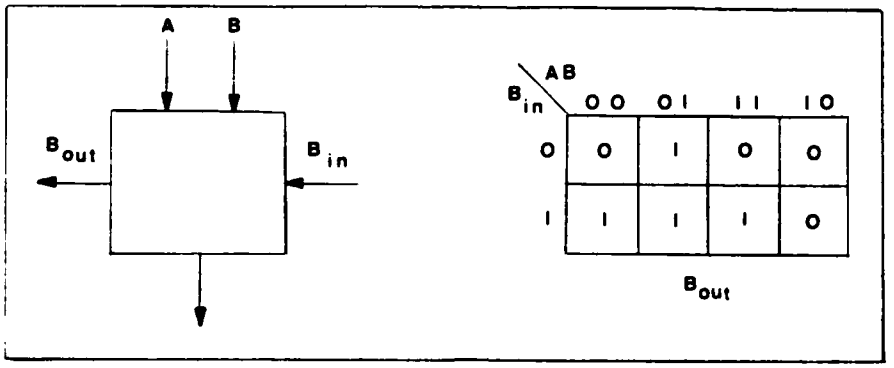


Figure 2.8 1-bit Subtractor and K-map of the Borrow Out

the Borrow generator and its K-map is shown in Figure 2.8. Here the auxiliary variable C_i is changed to B_{in} and C_{i+1} to B_{out} which denote Borrow in and out respectively. The boolean logic function is:

$$B_{out} = \bar{A}B + B_{in}(\bar{A} + B) \quad (2.7)$$

$$= B(A \oplus B) + B_{in}\overline{(A \oplus B)} \quad (2.8)$$

The expression (2.8) paves the way for an efficient implementation in VLSI CMOS technology. The use of this equation will be described later in Section 3.3.

2.5.4. Match Logic

In the last two sections, we discussed equality and magnitude comparison. If we place both the equality indicator and magnitude detector in each bit of the CAM array we have two output signals from the array for each word. They are Borrow (B_{out}) and Equal (E) signals. With these two signals, we can determine many magnitude relations between

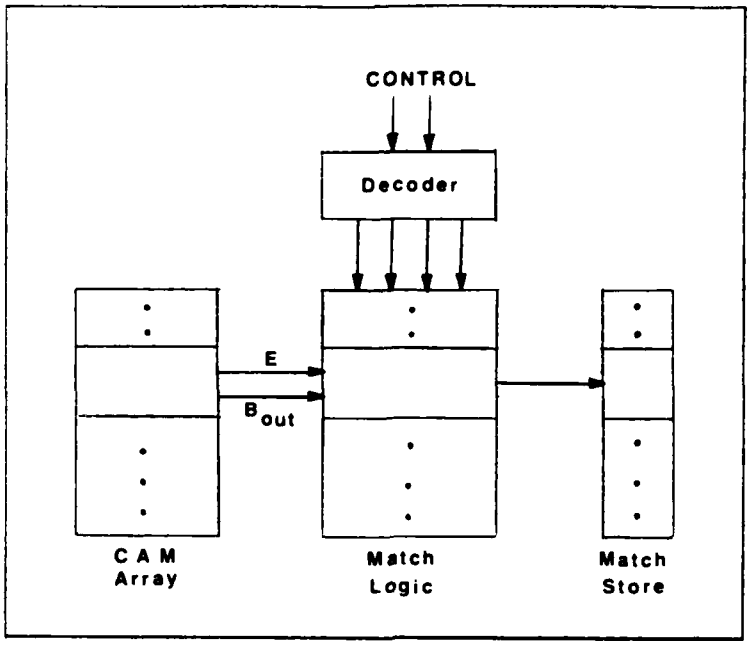


Figure 2.9 Match Logic Inputs and Outputs

the search argument (B) and the stored data (A) in the CAM. To generate the desired matches we need a match logic circuit between the CAM array and match storage (see Fig 2.9). The control signals A = B, A ≠ B, ..., etc, are sent to all the match logic cells from the decoder. The control signals must be applied externally through the I/O pins. In the AFIT CAM design, we use a 2:4 decoder to generate four control signals. These signals are A = B, A ≠ B, A ≥ B and A < B. The match logic circuit is shown in Figure 2.10.

We could have more magnitude relations if we use 3:8 decoder and more logic circuitry in the Match Logic. However the magnitude relations A > B and A ≤ B are more complex expressions than the ones we implemented in AFIT

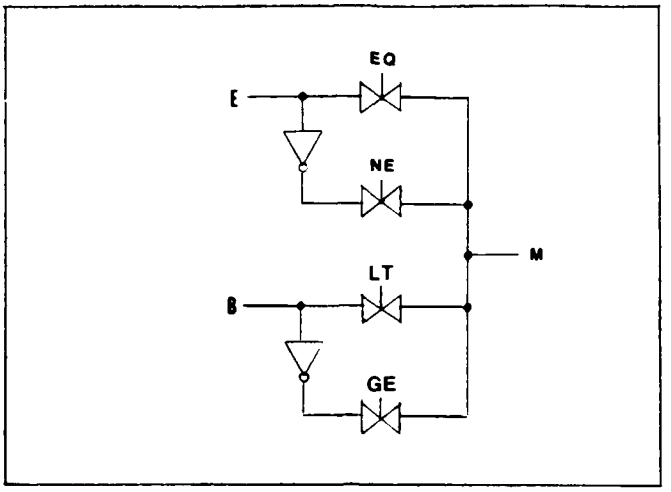


Figure 2.10 Match Logic Circuit

CAM. The implementation of additional magnitude relations in AFIT CAM is not possible due to lack of chip area. However it will be possible in next generation of AFIT CAM which should have up to a 25% reduction in the CAM cell size.

2.5.5. Handling Responses from the CAM Array

In a normal situation, there will be several matches resulting from operations discussed previously. The content of these words must then be read out one at a time. For this reason the CAM has a provision for feeding addresses of matches back into the CAM address decoder. However, there must exist some sort of queue server which is able to pick out a single match at a time, usually in a top-down order. The address readout operation will be repeated for all matching words. It is necessary to provide every M_i line of the Match Logic with a buffer flip-flop that can store the

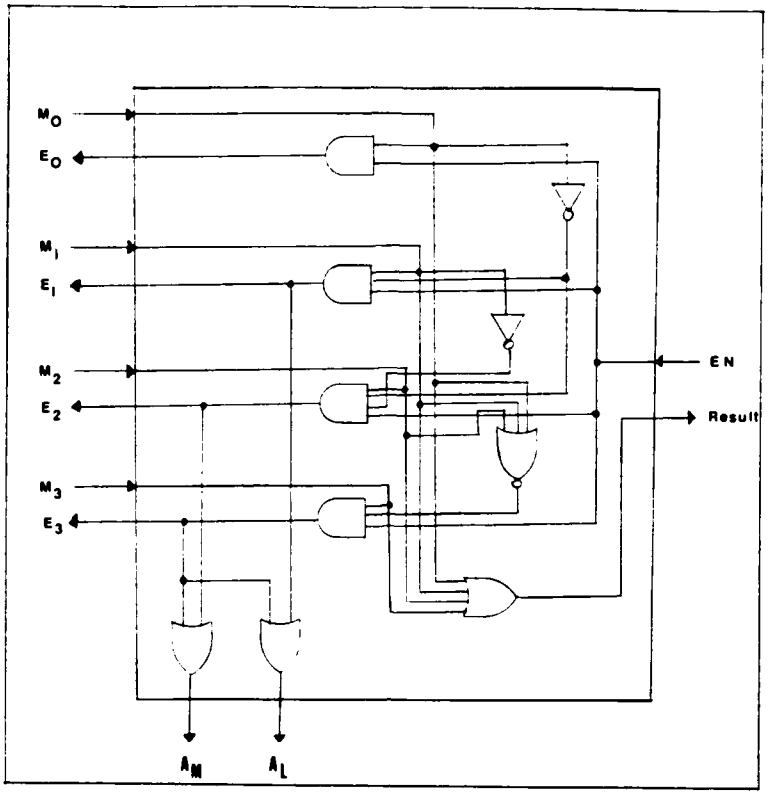


Figure 2.11 One module for a Multiple Match Resolver (22)

matches, a Match Store until the match is read out by the queue server (called a priority resolver).

The priority resolver outputs the address of the uppermost match. After reading out the uppermost match the active buffer flip-flop is reset. The next match then appears uppermost in the match store and can be served next.

The address encoder displays the uppermost matching address as shown in Figure 2.3.. The circuit which displays only the uppermost match and its address is called the multiple match resolver (MMR).

The priority resolver and the encoder can be replaced

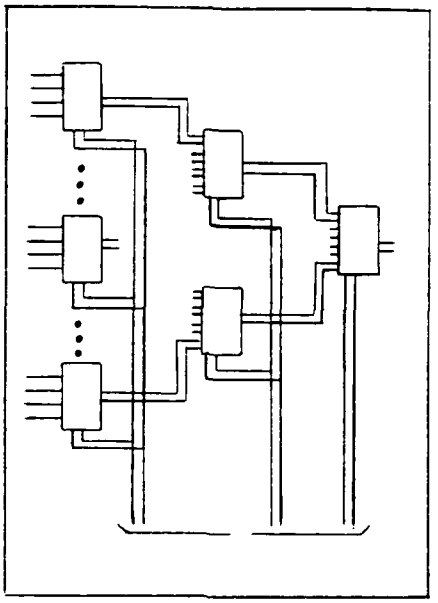


Figure 2.12 Multiple Match Resolver Modular Tree

with the Anderson's MMR (22). The cascaded gate structure in Figure 2.11 is intended to display only the uppermost match at its left-side output (E_i). By adding the two gates with wired-OR output to the basic block, shown in Figure 2.11, and connecting the blocks, as in Figure 2.12, the circuit will yield the correct address of the uppermost match. Note the tree structure which allows the uppermost match to be found in $O(\log_4 N)$ rather than $O(N)$ time where N is the number of words to be resolved.

Multiple matches can be handled very effectively by connecting MMR with the two master slave flip-flop (MSFF), as shown in Figure 2.13. The first MSFF stores the match non-destructively while the second temporarily stores the match until it is served. The LOAD signal will transfer the stored

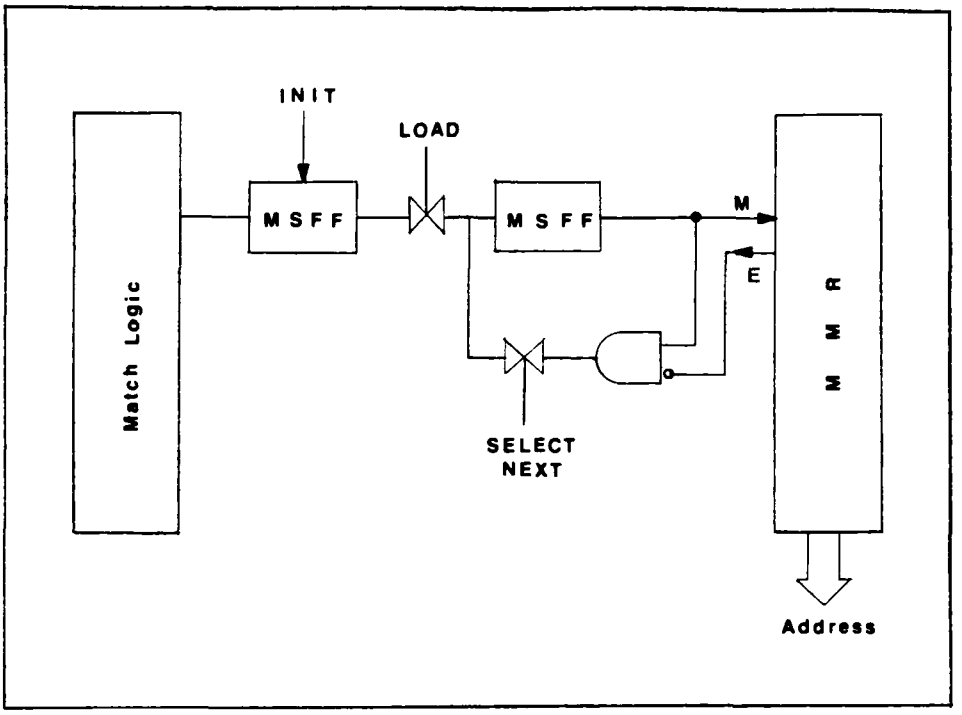


Figure 2.13 Handling of Multiple Matches

match in the first MSFF to the second MSFF without changing the contents of the first MSFF. The second MSFF is set or reset, according to the values of M and E, after a SELECT NEXT command turns on the transmission gate.

If the match line, M, shown in Figure 2.13 is "1" and is the uppermost match, then E becomes "1" due to the structure of the MMR. When SELECT NEXT turns on the transmission gate, the output of the AND gate (which is "0") will reset the second MSFF. The output of the second MSFF resets (M=0) and is no longer a match response. As long as M is "0", the MSFF can not be reset. E will not be a "1" until M is the uppermost match.

2.6. The Word-Parallel, Bit-Serial CAM

The memory structure presented in the previous section was for the all-parallel CAM. It performs a word-parallel and bit-parallel search for data. The basic organization of the word-parallel, bit-serial CAM will be discussed in this section. The advantages and disadvantages compared to the all-parallel CAM will also be highlighted.

2.6.1. Basic organization

The memory array in this CAM is similar to a typical RAM except that the decoder is used to address the bit-slice instead of the word. Additionally, comparison logic circuitry is not part of every CAM cell, unlike the all-parallel CAM. Since the search operation is accomplished serially, the comparison circuit can be placed outside the CAM array as shown in Figure 2.14. The memory array is made up of m words of n bits each. The bit-slice data can be read and written through the bit-slice register, to the left of the memory array, according to the bit-slice address. The searching is implemented simultaneously and in parallel, over all the words by reading the bit-slices one at a time. These outputs can now be compared with their respective bit of the external arguments. By the application of a sequence of bit-slice address codes, obtained from a bit-slice counter at the decoder inputs, all bit-slices can be made to appear at the memory output port. They can then be compared with their corresponding argument bits.

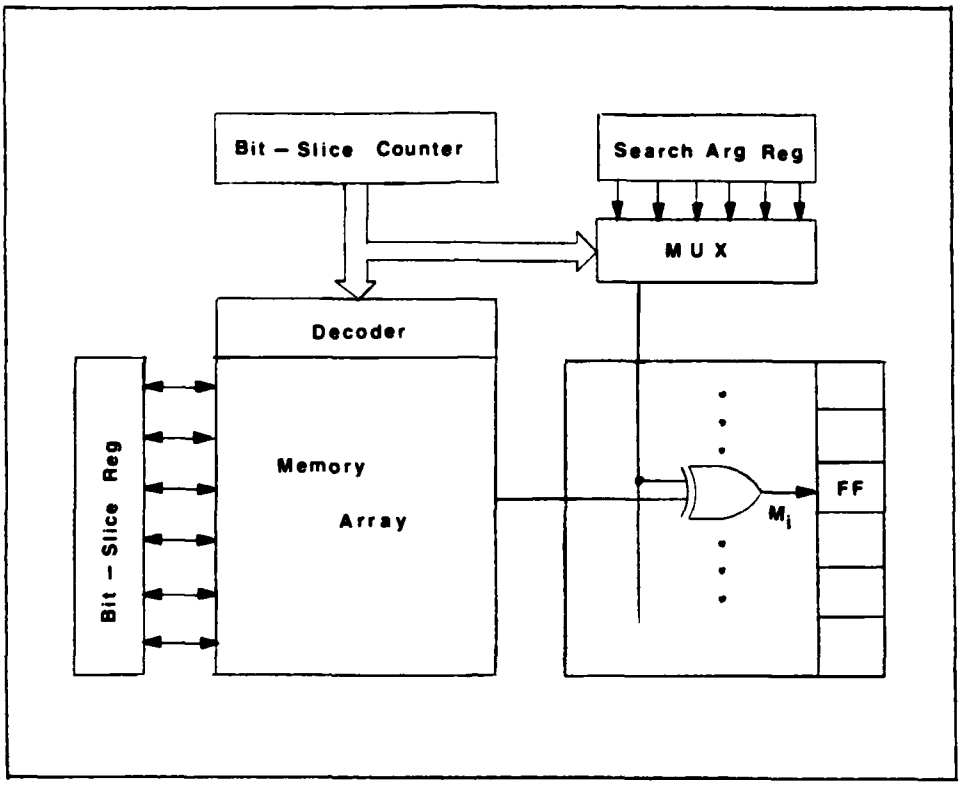


Figure 2.14. Organization of a Word-Parallel, Bit-Serial CAM

It is a simple task to pick out the argument bits, corresponding to the bit-slices, produced from the search argument register and made available for the comparison. This can be accomplished by a multiplexer controlled by the bit-slice counter. The address input into the decoder is the same bit position as in the argumemt register. If a comparison for equality were the only operation to be performed, only XOR gates are needed for each word location in the comparison logic circuitry as shown in Figure 2.14. We can have more complex comparison logic circuitry depending on the type of magnitude relations we desire.

Comparing the word with the search argument is done recursively in word-parallel, bit-serial CAM with the aid of comparison logic circuitry and the result storage component. The result storage circuits consist of a set of flip-flops (FFs), one for every word location. These are set to 1 before the search begins. As the search starts each flip-flop is set by an XOR gate output which obtains one of its inputs from the memory output and the other from the search argument register. A flip-flop in the result storage section is reset whenever a bit value, read from the memory array, disagrees with the bit value of the search argument (when $M_i = 1$). Once the flip-flop is reset, it must stay in the reset state even if M_i is set to "0".

2.6.2. Advantages and Disadvantages

As we can see from the above discussions, searching with this type of CAM is n times slower than the all-parallel CAM. It is much faster than a linear select RAM which requires m cycles for a linear search if $m >> n$. The word-parallel, bit-serial CAM is not as expensive as the all-parallel CAM due to less area being deducted to comparison logic, one column of logic instead of in every cell. Another advantage is the ability to read the word bits for comparison in any order. This is done by application of a corresponding address sequence to the decoder and multiplexer inputs allowing the bit counter to be replace by a more complex logic circuit. Futhermore, it is possible to select any subset of the bit

positions for a comparison. This is then equivalent to the masking operation.

Another drawback is this CAM's method for updating data, especially the alteration of a single word. This is not possible in a memory of this kind without rewriting the entire array. Addressed reading of a word location, to uncover its contents is an impossible operation using this basic design when only part of the word matches the search argument. This problem can be resolved if we add the ability of writing and reading these words with the penalty of consuming more chip area.

Because of its drawbacks, this serial CAM is not used for computing or control operations and is only suitable for simple table look-up or catalog memory operations as was presented earlier in Figure 2.1.

2.7. Using CAMs for More Complex Searching

In order to highlight some of the abilites of CAM and compare the characteristics of two different CAMs, the all-parallel and the word-parallel, bit-serial CAM, we will examine a complex search algorithm, to find the largest word stored in the CAM.

2.7.1. Ues of all-parallel CAM

For a maximum search using the all-parallel CAM, the initial contents of the mask and argument registers are as shown Figure 2.15. The mask register has a "1" in the

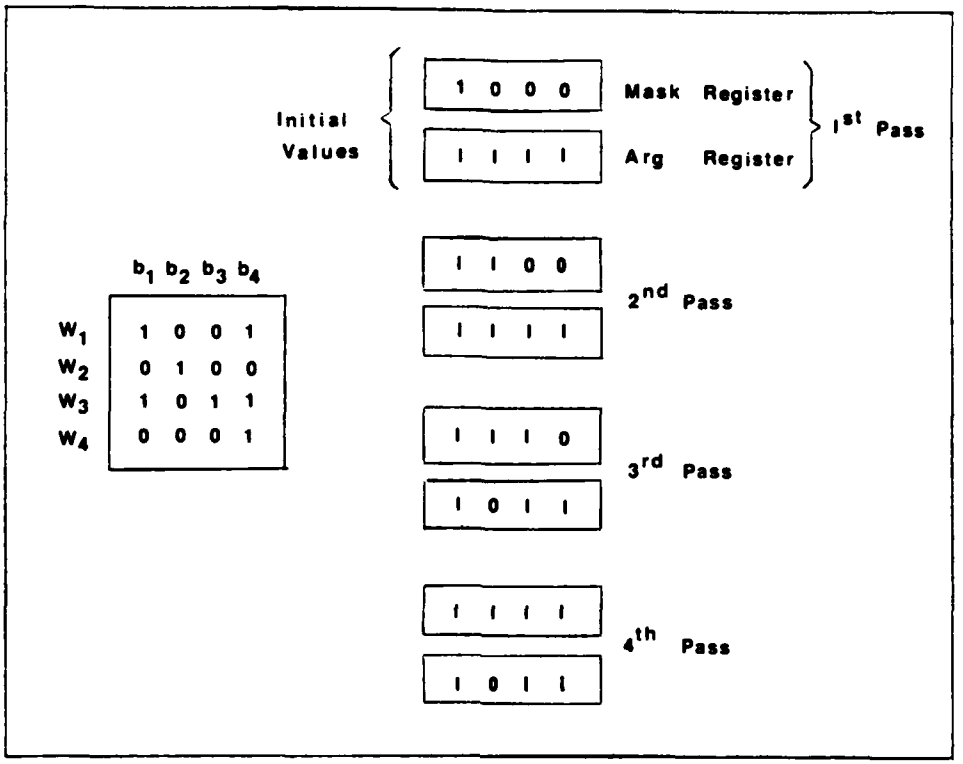


Figure 2.15 Maximum SearchUsing All-Parallel CAM

left-most position (the most significant bit) and the other bits are zeros. The argument register contains all ones. At the end of the search the largest number will reside in the argument register and the mask register will contain all ones. Note that this algorithm works only for unsigned numbers, not 2's complement numbers.

Now we test to see if there are any cells with leading ones. If a match occurs, the next search will proceed after setting the second bit of the mask register to one. If there is no match, the first bit of the argument register is set to zero. This will continue until all the bits have been

examined.

As shown in the example of Figure 2.15 the memory contains four words, each word having four bits. With the initial values shown in the figure, the search begins. In the first pass, all the bits are masked out except the left-most bit. Only the most significant bits in all the words are compared. There is at least one match in first pass so the argument register is updated. The words W_1 and W_3 are now candidates for being the largest number. In preparation for the second pass, the first and second bits are unmasked. This time, there is no match and the second bit of the argument register is set to zero. W_1 and W_3 are still candidates and the greatest number is now 10XX. In the third pass, we found a match. The greatest number must be 101X, therefore, W_3 is the greatest number. In order to find out the last bits value, we go through a final pass. The last bit is then determined and the largest number, 1011, is in the argument register, thus terminating the search.

2.7.2. Use of the Word-Parallel, Bit-Serial CAM

The basic idea for finding the greatest or smallest number using this CAM is similar to the method using the all-parallel CAM, The search is accomplished via the most significant bit check in a serial manner. The CAM considered for this search was shown in Fig 2.14.

Just as before, initially all the bits in the argument register to ones. After each pass, any unmatched bits will

reset the flip-flops, which were initially set to "1". Once reset the flip-flops must stay in this state. If there is no match (all words have zeros in bit position being probed), it is detected in the output circuit and all the flip-flops are prevented from reseting during this pass. This process is repeated until all the bits of the search argument have been exhausted.

Comparing the performance of the two CAMS in the search for the greatest number, it seems that the word-parallel, bit-serial CAM is faster than all-parallel CAM and easier to implement. It is however, very difficult to read the values of the greatest number from the word-parallel, bit-serial CAM even though we know where the greatest number is.

In most search algorithms, the equality and magnitude searches are most frequently used. For these searches, all-parallel CAM is n times faster than the word-parallel, bit-serial CAM, where n is number of bits in a word. Searching for a greatest number is not used very frequently. However, it can be done using a all-parallel CAM because of its masking capability.

There are many more algorithms to search for more complex specifications. These are the between-limits search, the search for the nearest-below or nearest-above, a search for the absolute-nearest, a proximity search, and seaches for boolean functions (1:161).

2.8. Use of CAPP for a Complex Search

In artificial intelligence, the term "problem solving" and "search" refer to a large body of ideas that deal with deduction, inference, planning, common sense reasoning, theorem proving, and other related processes. Application of these ideas is found in programs for natural language understanding, information retrieval, automatic programming, robotics, games, expert systems, as well as proving algorithms for mathematical theorems (11:21). The techniques for efficient searching are used in some of the areas of aritificial intelligence. In this section we examine one such search algorithm by using a content addressable parallel processor (CAPP).

In a large number of appications, we need to know whether point A is connected to point B directly or through a chain of intermediaries (23). Finding a path in a directed graph or determining the implications of an artificial intelligence semantic net are only two examples of such searches. Figure 2.16 shows a typical small digraph and its from-to list. There are usually many paths from a given starting position to a goal position when th length of that path is not important. Some of these methods are the depth-first search, the breadth-first search, the hill climbing, the beam search, and the best-first search. In the depth-first search, we search a descendant of a node before we explore other peers of that node. In the breadth-first search, the reverse is true. All the other searchs are

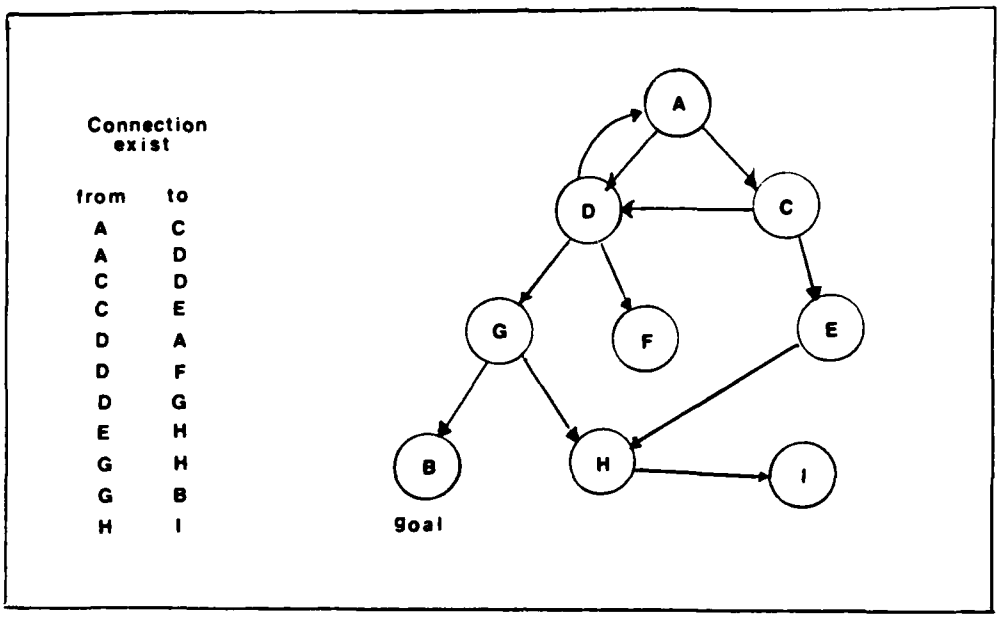


Figure 2.16 A typical Small Digraph

slightly more complicated than these first two methods (10:89-101). In depth-first searching, we would find a path to the destination node B to be A-C-E-H-I-H-E-C-D-F-D-G-B while a breadth-first search would produce:

A

C D

E D* F G A*

H H B

Where and * indicates that the node has already been visited. Using a CAPP we will do a breadth-first search, leaving a trail so that we can retrace our path from the terminal node back to the initial node. We will store information in the CAPP, using four fields, the source, destination, counter and

flag fields. One cell will exist for each connection of the graph. Initially all counter and flag fields will be set to zero.

We start at node A and select all cells which have this node as a source. The counter for each of these cells is marked with a one. A test is made to see if any cell with counter set to "1" has node B the target node as destination. If not, we then select the first cell with a counter set to one with its flag still zero. The flag is marked, set to one, to indicate that it has been processed, and its destination is read out. Using that destination as a source, we then select all cells with that source and counters set to zero. A two is then written into their counters. We repeat this for all cells with a counter of one. This gives the following memory snap shot for the digraph of Figure 2.16:

S	D	C	F
A,	C,	1,	1
A,	D,	1,	1

S	D	C	F
C,	D,	2,	0
C,	E,	2,	0
D,	A,	2,	0
D,	F,	2,	0
D,	G,	2,	0

Now we test to see if any second generation cells (count=2) have node B as a destination. If so, we are finished, otherwise, we select, in turn, cells with counters set to "2" and flags with "0" mark. Their descendants with initial counters zero are marked with a count of "3". The first cell of the second generation has D as a destination

node. Since D has already been visited, the flag has been set in all cells leading away from node D; no new activity is generated. After we have searched all the new descendants of the second generation, we go on to the third, and so on until the destination is reached. In this case the tracing of the net is complete after we have processed all cells of the second generation.

S	D	C	F
E,	H,	3,	0
G,	B,	3,	0
G,	H,	3,	0

Since B is among the destinations, shown above, we have reached our goal. We know a path does exist and in fact have found the shortest path. The remaining action is to read the path from the memory.

To trace the path, we begin at the end and work our way backward. B is the ultimate destination, so we look for the cell with G, B, 3, -. The link G-B is part of the path. Next, we look for a cell with G as a destination and a count field one less than the previous cells count, namely 2. The cell containing D, G,2, - is found. We now look for a cell with -, D, 1, -. This cell is A, D, 1, - and we are back to the starting node. The path has been traced back to A-D-G-B.

Suppose instead that H had been selected as the terminal node. When we ask for links entering node H we find cells containing E, H, 3, - and also G, H, 3, -. Since we are looking for the shortest path we will naturally choose the

link of the youngest generation (i.e., the one with the smallest count field). If, as in this case, two or more links have equal count we choose between them arbitrarily.

The algorithm we presented here is probably not a great deal faster than what which would run on a conventional machine. This is because all the generations of cells must be treated sequentially. If we mark all of the "outlinks" for a given node, in parallel, we could achieve a speed-up factor of two or three depending on the number of "outlinks" belonging to an average node. Where we really gain speed is in the algorithm used to read out the path. By storing the links in a CAPP, we now have implicitly stored the back links used by the algorithm. Without building a complicated tree structure when numbering the net, or without performing cumbersome searches, it is difficult to see how the algorithm presented could be implemented on conventional Von Neumann machines.

3. VLSI Implementation of AFIT CAM

3.1 Overview

We examined the basic organizations of different types of CAMs in Chapter 2. Advantages and disadvantages were also highlighted. Since we need a very high performance general purpose CAM, the all-parallel CAM is the best choice to implement in hardware. This CAM is very effective for use in searching and magnitude comparison algorithms. We can also access data easily with the all-parallel CAM. The all-parallel CAM however consumes more chip area because it requires comparison logic circuitry to be placed in each CAM cell.

The AFIT CAM is designed to support specific needs. First, it must support the set of operations for image analysis. In order to accomplish this task, we need a parallel writing capability. Second, the AFIT CAM must have the ability to reset the 16th bit in every memory cell based on a match result from a comparison to support garbage collection. Storing and retrieving data must be easily accomplished as in a conventional memory which has direct access to load and unload the memory. Finally, it must should provide some support for general purpose computing.

With these specific requirements, the CAM to be implemented is much more complicated than the basic CAM described in the Chapter 2. However, the basic overall concepts and structures are still similar.

The AFIT CAM is implemented in 3u CMOS technology. The chip is logically organized as 128 words of 16 bits. In this chapter we will discuss the hardware implementation of AFIT CAM. At the end of the chapter we will present the floor plan and functional chararcteristics of AFIT CAM.

3.2. Organization of AFIT CAM

The block diagram of the AFIT CAM system is shown in Figure 3.1. Its key sections are the Argument Register, Data Register, Mask Register, CAM Array, Match Logic, Match Store, Auxiliary Logic, Temporary Match Store, Multiple Match Resolver (MMR) and some other logic circuits needed for interconnection and control.

The registers consist of 16 master slave flip-flops which can be loaded from the data bus in parallel. Their outputs are sent to the CAM Array and other logic circuitry.

As stated previously, the CAM Array is a 128 by 16 array of CAM cells. Each CAM cell consists of a 6-transistor CMOS static memory cell, a magnitude detector and an equality indicator. The Borrow and Equal signals from the array are sent to the Match Logic section.

The Match Logic selects matches according to the magnitude relationship control signals. These control signals are Equal (EQ), Not Equal (NE), Greater Than or Equal (GE) and Less Than (LT).

The registers in the Match Store section can be set, or reset, according to the previous match stored there and to

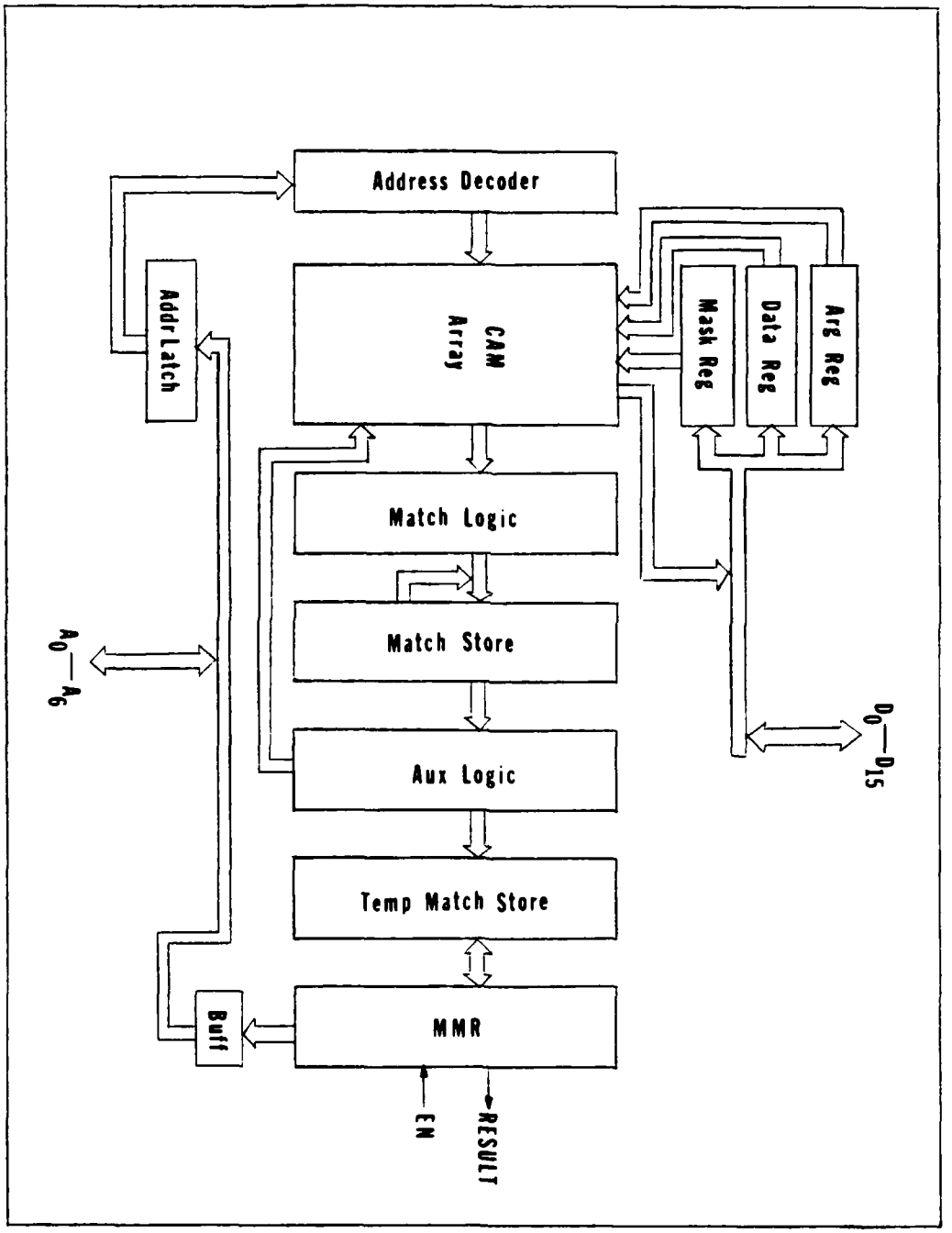


Figure 3.1 Organization of AFIT CAM

the new match signal received from the Match Logic. This allows us to perform a more complex search than the simple one previously described.

The Auxiliary Logic section takes the match signal, from the Match Store and other control signals, resets the 16th memory cell and drives the wordline during parallel writing. The forwarding of the match signal to the Temporary Match Store section is also controlled by the auxiliary logic.

The match information stored in the Temporary Match Store is destructive. The match information will be reset after it has been serviced by the MMR.

The MMR generates the uppermost match address. The address is available at the bottom of MMR. If there are mutiple matches in the Temporary Match Store, then it resets the uppermost match when the Select Next control signal is initiated. After this, it automatically outputs the next uppermost match address. If there is any match, it is indicated by the RESULT output. The MMR can be enabled or disabled by the enable input EN. The match address produced, by the MMR, can be sent to the address I/O pad or can be latched into the Address Latch section. The latched address is finally fed into the address decoder.

The AFIT CAM design can be divided into three major parts. They are the CAM cell, RAM circuitry, and the CAM circuitry. These three parts are discussed in detail in next three sections.

3.3. CAM Cell

The CAM cell is made up of data storage elements and magnitude comparison logic circuitry. The six-transistor CMOS static RAM is used for data storage for AFIT CAM. The operation and the logic of the six-transistor static RAM are the same as discribed in Chapter 2. However, the peripheral devices for the RAM circuitry are slightly different.

The size of the CAM cell is very important to the overall chip area. This cell must be iterated 128 by 16 times to construct a 2K CAM. The CAM array consumes well over half of the available chip area.

In Chapter 2 we developed the Karnaugh map for the Borrow Out signal of the 1-bit subtractor which is used as the general cell for the 16-bit subtractor. The magnitude detector and the equality indicator can be implemented by using Eq. (2.8) from Section 2.5.3. The logic diagram implementing eq. (2.8), $B_{out} = b_i (a_i \oplus b_i) + B_{in}\overline{(a_i \oplus b_i)}$, is shown in Figure 3.2. Here we use an XOR gate, two transmission gates and an inverter for the magnitude detector. Two transistors are also used for the equality indicator. This requires a total of 12 transistors. If we implement the magnitude detector using Eq. (2.7), it requires a total of 12 devices. We need 4 devices for the XOR gate and 2 devices to construct the equality indicator. In this case, we need a total of 18 transistors. A savings of six transistors is realized by using Eq. (2.8). Since there are 2048 CAM cells (128 words by 16 bits), we save 12,288

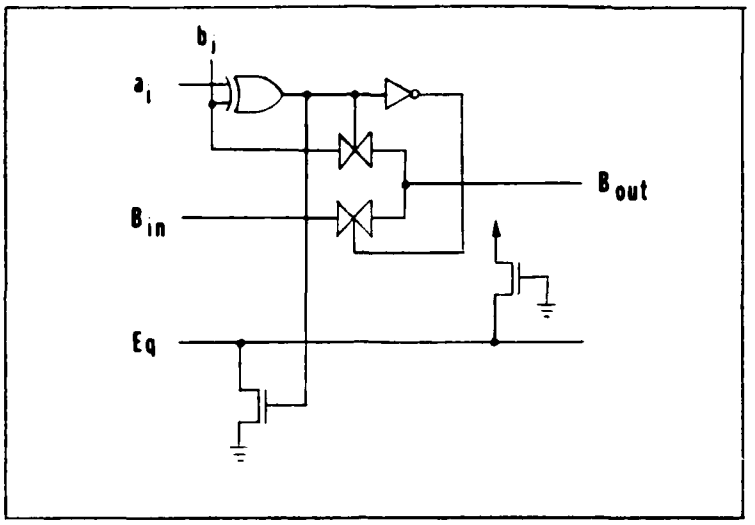


Figure 3.2 Magnitude Detector and Equality Indicator Without Masking Capability

transistors for the CAM chip. This is a significant savings.

The circuit shown in Figure 3.2 does not have the masking capability. The masking capability can be added to the above circuit by using Eq. (2.3) from section 2.5.2. When a paticular bit is masked, the Borrow (B_{in}) and the Equal (Eq) signals must be forwarded to the next stage without computation. The complete logic diagram for the magnitude detector and the equality indicator with the masking capability is shown in Figure 3.3. The transistor level diagram for Figure 3.3 is shown in Figure 3.4. If we repeat this cell 16 times we build the 16-bit magnitude detector and the equality indicator. The pullup p-device for the equality indicator shown in Figure 3.3 is placed after the 16th cell. Thus, the Borrow Out signal to the last bit

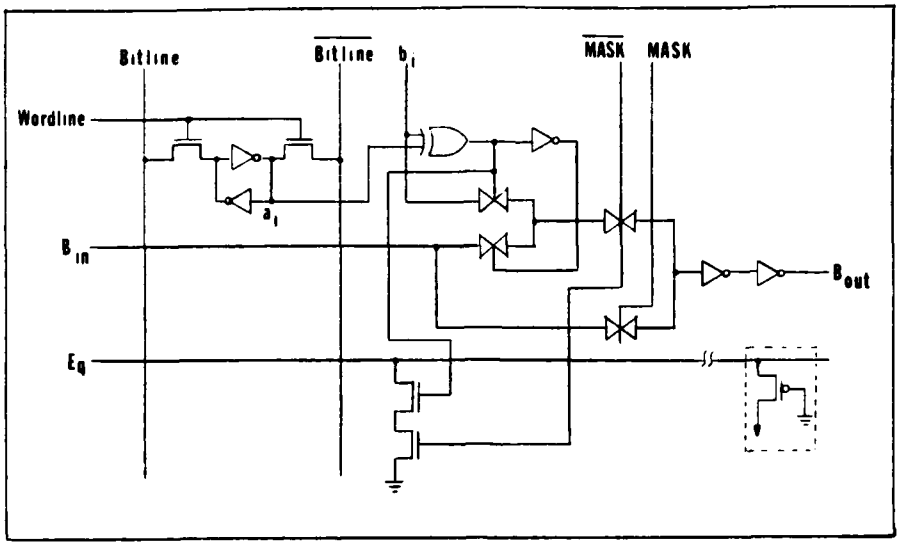


Figure 3.3 AFIT CAM Cell

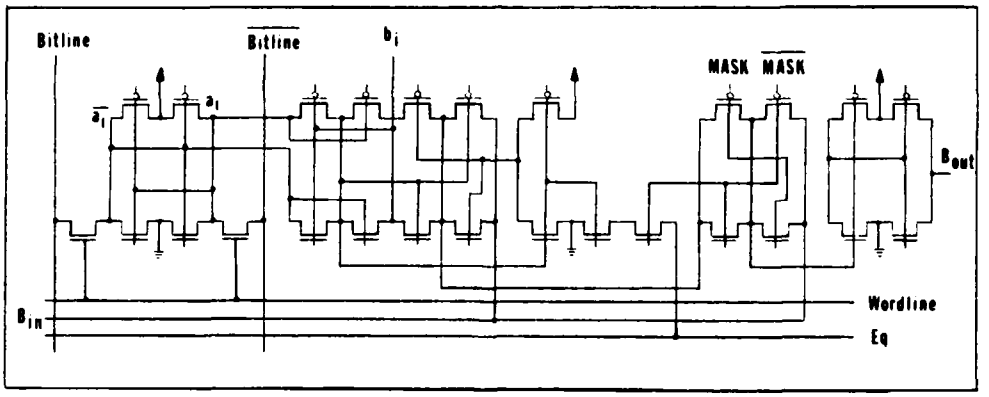


Figure 3.4 Transistor Level Diagram for AFIT CAM

is much slower than the equal signal. This is because the borrow signal must be propagated through all 16 stages. In order to forward the borrow out signal to the next stage, the signal must propagate through a 2-to-1 multiplexer and two invertesr. When any of the stored data bits disagrees with the unmasked argument bits the equal line is pulled low

("0"), otherwise the Equal line is high ("1"). The equality indicator uses wired-OR logic and hence is much faster than the Borrow signal. A SPICE circuit simulation shows that the worst case propagation delay of the Borrow signal is 160 ns while the Equal signal has a worst case delay of 28 ns.

The stored bit a_i and $\overline{a_i}$ are taken from the memory cell as shown in Figure 3.4. The CMOS XOR gate needs both the a_i and $\overline{a_i}$. The argument bit (b_i) is broadcast from the argument register. The MASK and $\overline{MASK}$ are broadcast from the mask register. Both registers are situated above the CAM cell so that broadcasting is accomplished vertically from top to bottom. The masking operation of the argument register is actually done in the CAM cell rather than by the registers.

3.4 RAM Circuitry

The block diagram of the random access memory (RAM) circuitry, with peripherals, is shown in Figure 3.5. The RAM circuitry consists of the address decoder, a wordline driver, and the data and mask registers along with logic circuitry necessary for reading from and writing to the memory cell.

To read from the memory cell, the three p-devices at the bottom of the figure are used with a sense amplifier. Both bitlines are precharged to V_{dd} by turning transistors P4 and P5 off. Since the sense amplifier takes the difference between the voltages of the two bitlines, we want to have both bitlines at the same potential. Transistor P3 is turn on for this purpose (Ref 17). A two stage CMOS differential

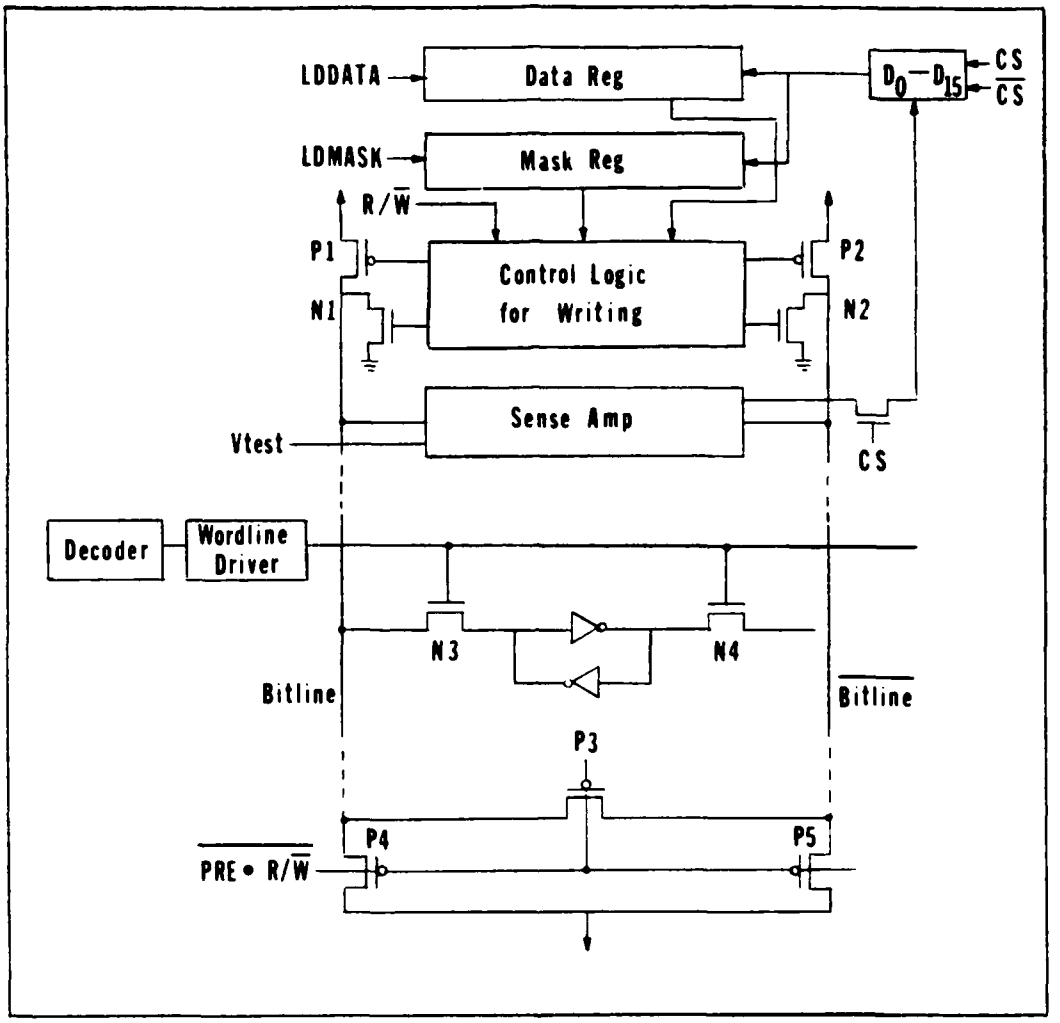


Figure 3.5. RAM Parts of AFIT CAM

amplifier is used as the sense amplifier (Ref 18).

Finally, to write to the memory cell, four transistors P1, N1, P2 and N2 are used. These four transistors are controlled by the logic circuitry as described in the "Control Logic for Writing" section.

3.4.1 Address Decoder

The address decoder will select a wordline according to

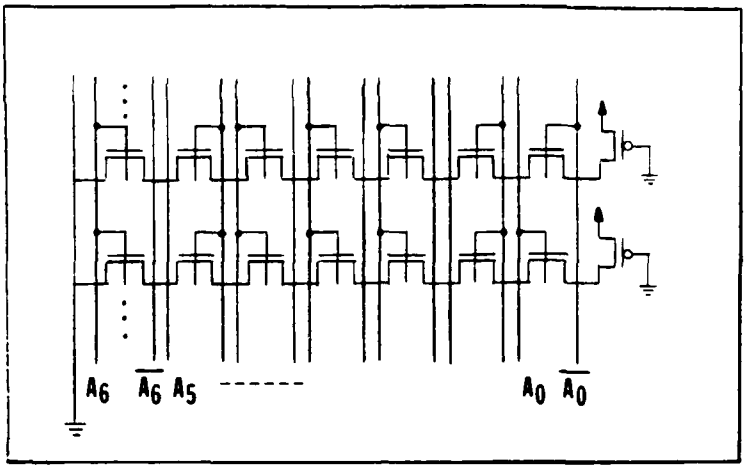


Figure 3.6 Address Decoder

the address signals A_6 through A_0 as shown in Figure 3.6. The address latches broadcast each bit of the address and its complement signal all the way up to the top of the CAM Array. The gates of the n-devices are connected to the proper true or complement address lines. For a example, address 1011100 and next address 1011101 are shown on the figure. The gates of the n-devices in address 1011100 are simply connected to the vertically broadcasted address lines $A_6\overline{A_5}A_4A_3A_2\overline{A_1}\overline{A_0}$. When all the gates of n-devices detect all ones the output will be zero otherwise the pull-up device gives an output of one. The outputs are inverted in the wordline dirver to

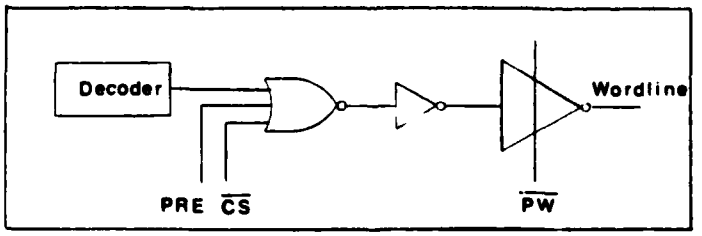


Figure 3.7. Wordline Driver

give a true wordline indication.

3.4.2 Wordline Driver

The wordline controls all the pass transistors in the memory cells to read or write. The wordline must be driven horizontally to control all the pass transistors in the memory cells, so that all 16 bits of a word can be accessed simultaneously. The wordlines must also be controlled by various signals as shown in Figure 3.7. These control signals are PRE, CS and the output signal from the address decoder. The control signal sent to the large inverter input, PW, is used to prevent a conflict driving the wordline during a parallel writing. This inverter is turned on only when there is no parallel write operation in progress (PW = 0). When a parallel write is being performed the wordlines are driven from the other side of the CAM cell.

3.4.3. Sense Amplifier

A one stage differential amplifier with current mirror active loading was discussed in the Chapter 2. In this section, we will discuss the amplifier in greater detail. Also, we will present the two stage differential amplifier used for actual implementation of the sense amplifier in the AFIT CAM.

A one stage differential amplifier is shown in Figure 3.8. The voltage gain of the one stage differential amplifier is

$$A_{v1} = \frac{V_{out}}{V_{in}} = -G_{m1}\left(r_{o2} \,//\, r_{o4}\right)$$

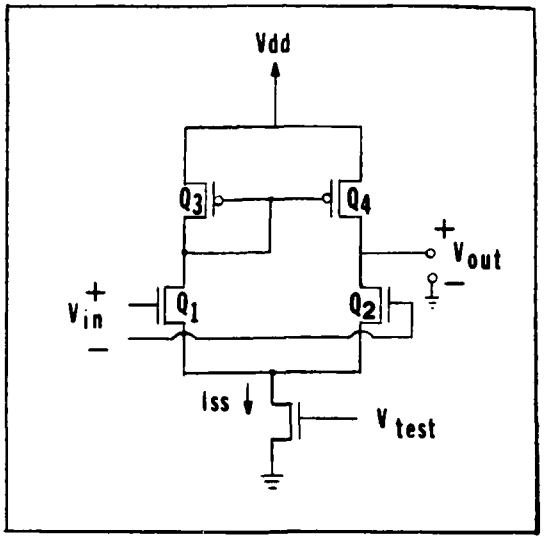


Figure 3.8 One Stage Differential Amplifier

where

$$G_{m1} = \mu_s C_{ox} \left(\frac{W}{L}\right)(V_{gs} - V_{th}) \quad (5:265)$$

$$g_{o2} = I_{D2}/V_{A2}$$

$$g_{o4} = I_{D4}/V_{A4}$$

g_{02} and g_{04} are output conductance of Q_2 and Q_4 respectively. The transconductance Gm can also be written $2*I_{ss}/(V_{gs}-V_{th})$ since $I_{ss} = \left[(\mu_s C_{ox})/2\right](W/L)(V_{gs}-V_{th})^2$. V_{A2} and V_{A4} are the channel length modulation voltages for Q_2 and Q_4. I_{d2} and I_{d4} are the drain currents of Q_2 and Q_4. All other terms are as defined in Section 2.5.1.

After substituting Gm, r_{02} and r_{04}, the equation for A_{V1} becomes

$$A_{V1} = \frac{-2 I_{ss}/(V_{gs}-V_{th})}{\left(\frac{I_{D2}}{V_{A2}} + \frac{I_{D4}}{V_{A4}}\right)} = \frac{-4\left(\frac{1}{V_{gs}-V_{th}}\right)}{\left(\frac{1}{V_{A2}} + \frac{1}{V_{A4}}\right)}$$

since the currents through Q_2 and Q_4 are the same and are equal to half of I_{ss}.

We note further that the voltage gain is inversely proportional to $(V_{gs}-V_{th})$.

Since $I_d = [(usCox)/2](W/L)(V_{gs}/V_{th})^2$, A_{V1} can be rewritten again as

$$A_{V1} = \frac{-4\sqrt{\frac{[(\mu_c C_{ox})/2](W/L)}{I_D}}}{\left(\frac{1}{V_{A2}} + \frac{1}{V_{A4}}\right)}$$

The voltage gain will be inversely proportional to $\sqrt{I_D} = \sqrt{I_{ss}/2}$. And the low-frequency voltage gain can be increased by operating the amplifier at a lower quiescent current level. Therefore, V_{gs} must be low in order to have this low quiescent current. The current-voltage characteristics of the *enhancement* MOSFET show that $\Delta I_D/\Delta V_{DS}$ is small with low values of V_{gs}, so we can construct a constant current source (I_{ss}).

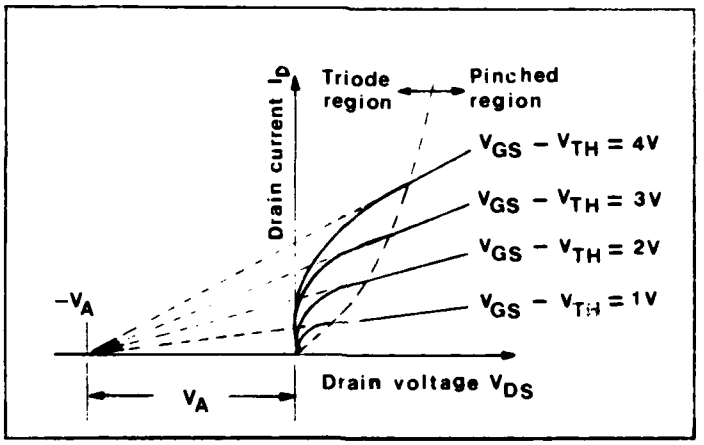


Figure 3.9. Actual Current-Voltage Charateristic of *n*-Channel Enhancement-mode MOSFET Showing the Effects of Channel Length Modulation

The actual current-voltage characteristics of a n-channel enhancement MOSFET showing the effects of channel length modulation are shown in Figure 3.9. The channel length modulation effect will not be discussed.

The above discussion is true only when the amplifier is operated at low frequencies. If we consider the effect of a capacitive load (C_L) on the performance of this circuit, the gain equation becomes, (20:236),

$$A_{V_2} = G_{m_1}\left(r_{o_2} // r_{o_4} // \frac{1}{j\omega C_L}\right)$$

This can be rewritten as

$$A_{V_2} = \frac{G_{m_1}}{g_{o_2} + g_{o_4} + j\omega C_L}$$

From this we can see that A_{V2} decreases as the operating frequency increases.

A SPICE circuit simulation of the differential amplifier was run with the channel length (L) and width (W), of Q_1, Q_2, Q_3, and Q_4, set to 3um and 6um respectively. Q_5 was used as a current source and its channel length and width was 9um by 9um. The inverter was then connected to the output of the amplifier. The results of the SPICE simulation showed that the voltage gain decreased as V_{test} increased. The differential voltage input must be greater than 0.15V before the inverter can detect a bit value. This process is too slow and the voltage gain must be much greater for fast detection of the differential voltage on the bitlines.

An improved two stage cascode amplifier circuit as shown in Figure 3.10 was simulated with a SPICE program. In the

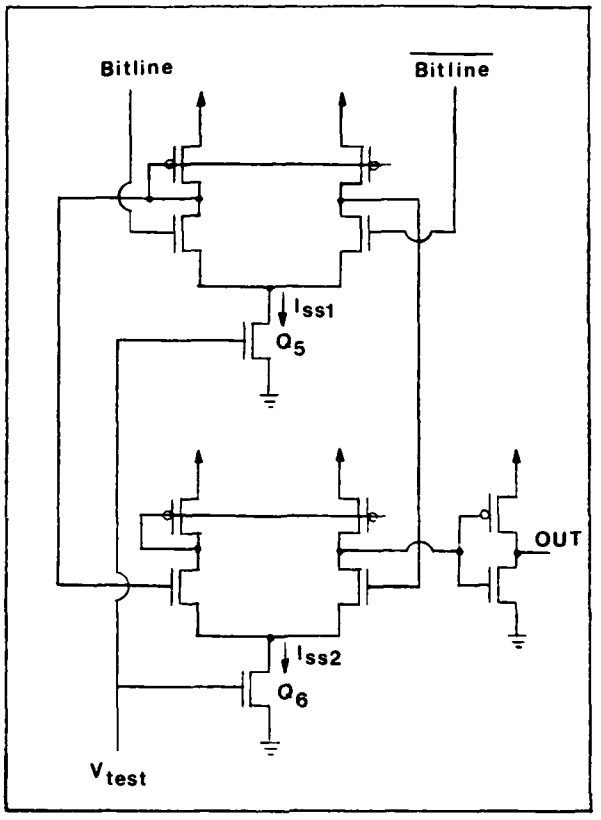


Figure 3.10 Two Stages Differential Sense Amplifier

simulation, the inverter detects the output when the differential voltage between Bitline and Bitline is 0.01 volt, providing much faster detection.

3.4.4. Control Logic for Writing

The ability to write to a particular bit or bits which are a subset of a word is very useful in some applications. This can be accomplished by using a "maskable write" feature incorporated into the AFIT CAM.

The word which will be written into the memory is stored in the data register. In a regular RAM, all of the bits in a

word are written simultaneously into the memory cells of the selected word. In the AFIT CAM, however, writing is conditioned on the mask register. If the mask bit position is a one, we write the data bit into the memory cell, otherwise the memory cell will not be changed.

Writing a zero into the memory cell is done by opening the pass transistor after precharging the bitline to V_{dd} and discharging $\overline{\text{bitline}}$ to ground. The two n-device pass transistors, N3 and N4, are then opened by the wordline as shown in Figure 3.5. Writing a one into the memory cell is the opposite of writing a zero. The memory cell cannot be written if both bitlines are precharged to V_{dd}, even though the pass transistors are open. When MASK is zero both bitlines are pulled to V_{dd}. Writing can only take place when the R/W input is zero. The truth table is shown in Table 3.1. This shows the control required for the four devices which drive the bitlines during a write operation.

When reading a memory cell ($R/\overline{W} = 1$) no transistors are opened as per Table 3.1. Both bit lines are precharged

Table 3.1. Truth Table for Control Logic for Writing

	R/W	Bit	Mask	gP1	gN1	gP2	gN2
Write	0	0	0	0	0	0	0
	0	0	1	0	0	1	1
	0	1	0	0	0	0	0
	0	1	1	1	1	0	0
Read	1	0	0	1	0	1	0
	1	0	1	1	0	1	0
	1	1	0	1	0	1	0
	1	1	1	1	0	1	0

to V_{dd} through the three transistors in the bottom of the circuit of Figure 3.5. The input signals gp1, gn1 ,gp2 and gn2 go to the gates for transistors P1, N2, P2 and N2 respectively. The logic equations for the gate signals are

$$gP1 = (R/\overline{W}) + (Bit \cdot Mask)$$

$$gN1 = \overline{(R/\overline{W})} \cdot Bit \cdot Mask$$

$$gP2 = (R/\overline{W}) + (\overline{Bit} \cdot Mask)$$

$$gN2 = (R/\overline{W})\ Bit \cdot Mask$$

Converting these equations for AND-OR logic to NAND-NOR logic we get the implementation in Figure 3.11.

The four devices P1, P2, N1 and N2 must be big enough to drive all 64 memory cells for parallel writing. Therefore, they are high current driving devices. The channel length (L) and width (W) of both P1 and P2 devices are 3u by 117u each. The size of N1 and N2 are each 3u by 70u each.

3.4.5. Data/Mask Registers

The Data Register holds the bits of information to be stored in the CAM on a write operation. Its output is sent to the writing control logic circuitry. The Mask Register stores the masking information and its outputs are also sent to the writing control logic circuit and to the maginitude comparison logic circuit inside each CAM cell. Each register stores 16 bits of information and can be loaded in parallel.

The logic diagram of both the data and the mask registers is shown in Figure 3.12. The MSFF can be loaded

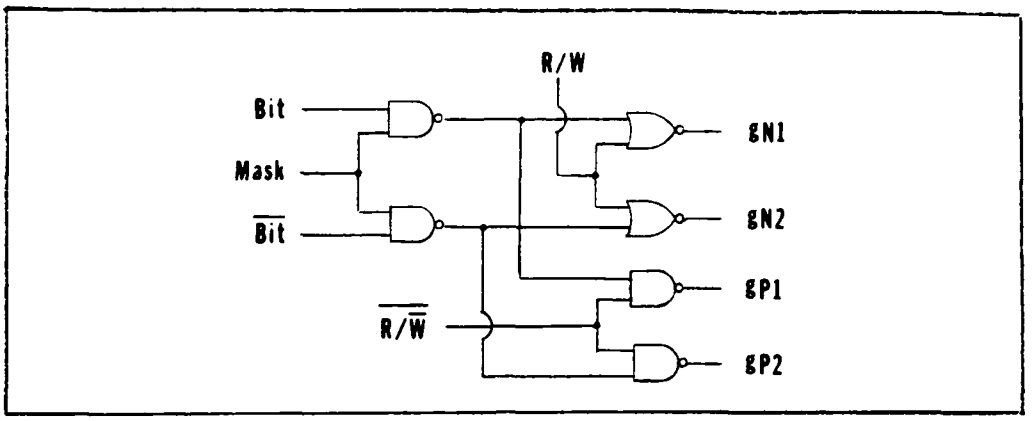


Figure 3.11 Control Logic for Writing

via the control signals LDDATA and LDMASK. The MSFF is comprised of 2 D flip-flops. The transistor level diagram of the flip-flop is given in Figure 3.13.

Note that the argument register is situated just above the data register. But it is not shown in Figure 3.12 because it is not part of the RAM circuitry. Its output goes to the magnitude comparison circuit in the CAM cell.

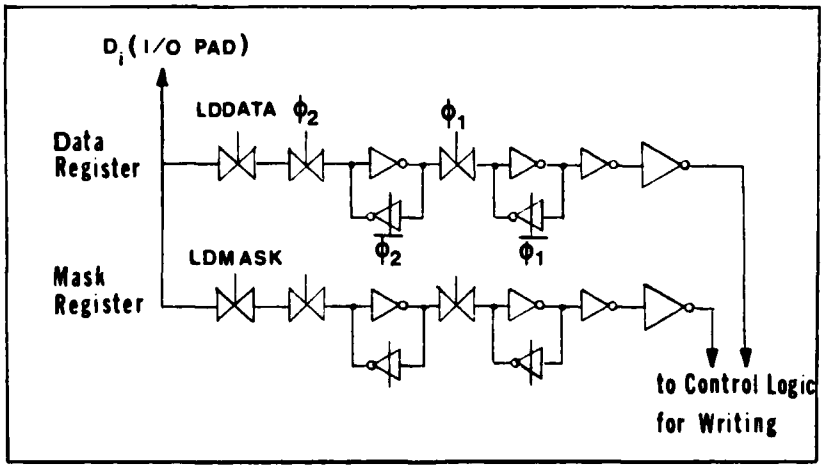


Figure 3.12. Data/Mask Registers

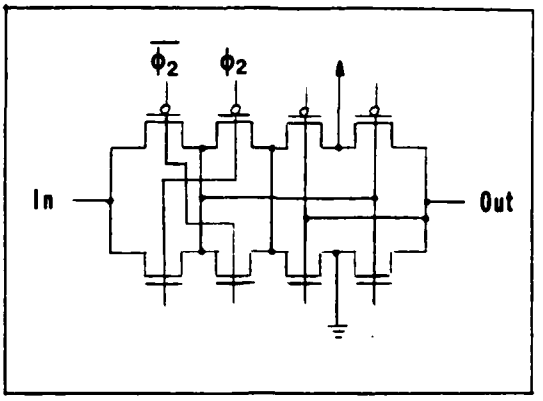


Figure 3.13 Flip-Flop

3.5. CAM Circuitry

The CAM circuitry consists of the magnitude detector, Match Logic circuitry, Match Store, Temporary Match Store, Multiple-Match Resolver and Auxiliary Logic Circuitry as shown in Figure 3.14.

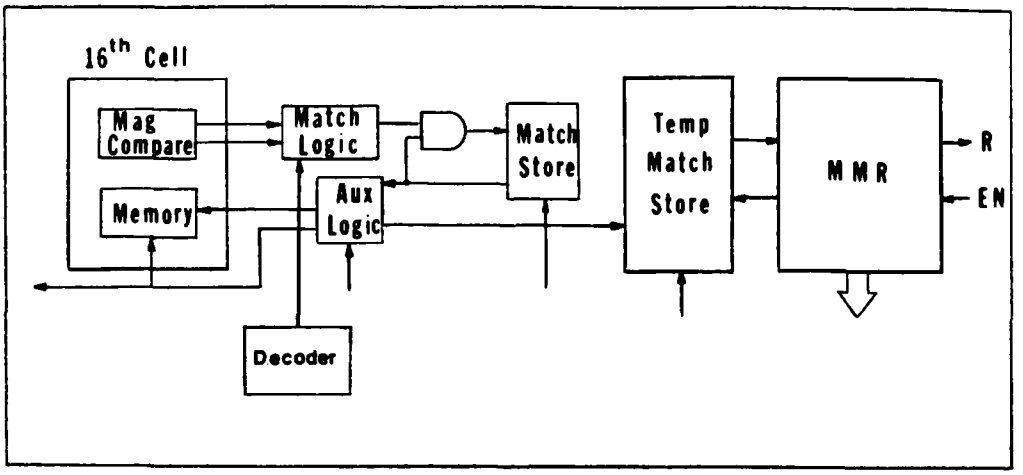


Figure 3.14 Block Diagram of CAM Parts

3.5.1. Match Logic

The Match Logic circuit was shown in Figure 2.10. In

the actual implementation of the AFIT CAM, we used pass transistors instead of transmission gates. Because a transmission gate requires both true and inverse control signals. This would cost us more circuitry to generate inverses control signals inside the Match Logic. The transistor level diagram for the match logic circuit is shown in Figure 3.15. The control signals come from the 2:4 decoder, and only one of the four signals is "1" at a given time. When EQ is "1", the Equal signal (E) is forwarded to inverter as a M. When NE = 1, the inverted signal of E ($\overline{E}$) is forwarded. The Borrow signal (B) and its inverting signal ($\overline{B}$) is forwarded to the inverter according to the control signals LT and GE. Since the next stage uses negative logic, the output is inverted. The inverter also provides the required strong output signal.

3.5.2. Match Store

The input signals to the Match Store are CONT, $\overline{\text{CONT}}$, and $\overline{\text{INIT}}$. They are driven from input pads on the AFIT CAM chip.

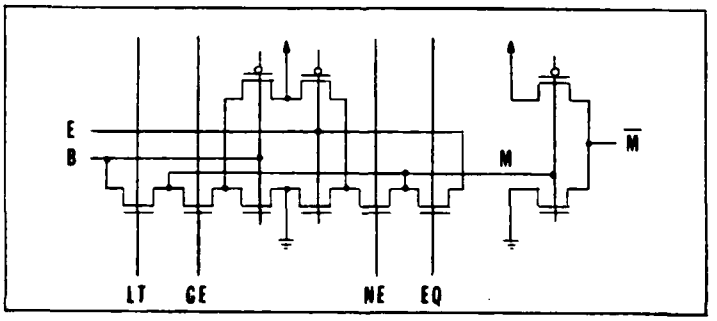


Figure 3.15 Match Logic

The $\overline{INIT}$ signal will reset the flip-flop to "1" as shown in Figure 3.16.

The flip-flop must be initialized before the search begins. Thus, initially words match. The next match results are ANDed with this flip-flop to store a new match signal. Finding all the stored numbers in the memory cells that are less than X and greater than or equal to Y ($X > A \geqslant Y$) can be done in 3 search cycles. First initialize Match Store flip-flops to one. Second we find all the numbers less than X.

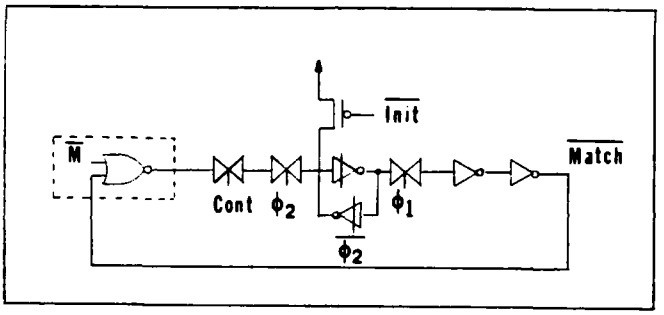


Figure 3.16 Match Store

This will reset the flip-flops that are not matched and those numbers are eliminated from futher consideration. Next we find all the numbers that are greater than or equal to Y. This will set all the unmatched flip-flops. The flip-flops which were not set during the two search cycles hold the match signals and are the numbers we are looking for.

3.5.3. Auxiliary Logic

The Auxiliary Logic circuitry generates the signal which resets the last memory cell column of the CAM array. This is

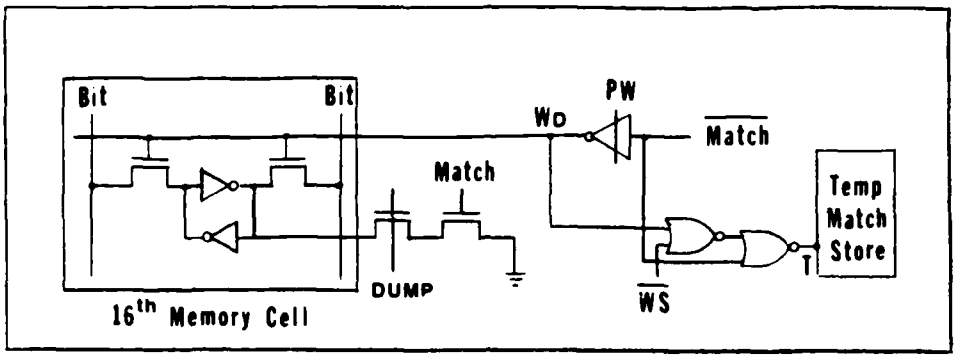


Figure 3.17 Auxiliary Logic

done when the are match and DUMP control signals set. This capability is very useful for garbage collection because last memory cell can be used as a vacancy indicator. The match signal is generated from the Match Store as in Figure 3.16. The gate level diagram for the 16th memory cell is shown in Figure 3.17. The wordline can be driven by this circuitry when the Parallel Write, PW, signal is set and there is match, PW=1 and MATCH=0. Parallel writing is accomplished in two cycles, 64-words at a time. There are two vertically broadcast parallel write lines, PW1 and PW2. This is implemented this way because in the worst case the precharge devices for both bitlines are not large enough to write all of the 128 words simultaneously.

The match signal is forwarded to the Temporary Match Store when output of the decoder is one (W = 1) or word selection is not in progress ($\overline{WS}$ = 1). Normally, WS is set at one, but we set WD = 1 and W = 0 when checking a single word matched across multiple chips.

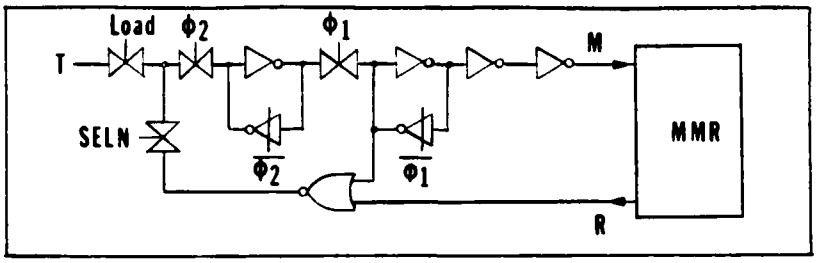


Figure 3.18 Temporary Match Store

3.5.4. Temporary Match Store

The match information stored in the Temporary Match Store is potentially destructive. After the uppermost match register is serviced, the register is reset. This logic circuitry, along with the MMR, forms the priority network. The logic diagram is shown in Figure 3.18.

The input signal T from the Auxiliary Logic circuitry is loaded into the MSFF by the LOAD control signal. The Temporary Match Store is loaded after the search is completed. If a register in the Temporary Match Store holds an uppermost match. then M=1 and E=1 and the rest of the match registers have M=1 and E=0. When Select Next (SELN) goes high the register will be reset, and M becomes zero. This triggers an output to the next uppermost match address at the bottom of MMR, and sets next the uppermost match enable signal, E, to one. Thus this becomes a uppermost match.

3.5.5. Multiple Match Resolver (MMR)

The MMR logic was explained in detail in the Chapter 2. In this section the VLSI implementation is presented.

A total of 43 modules are used in the AFIT CAM MMR

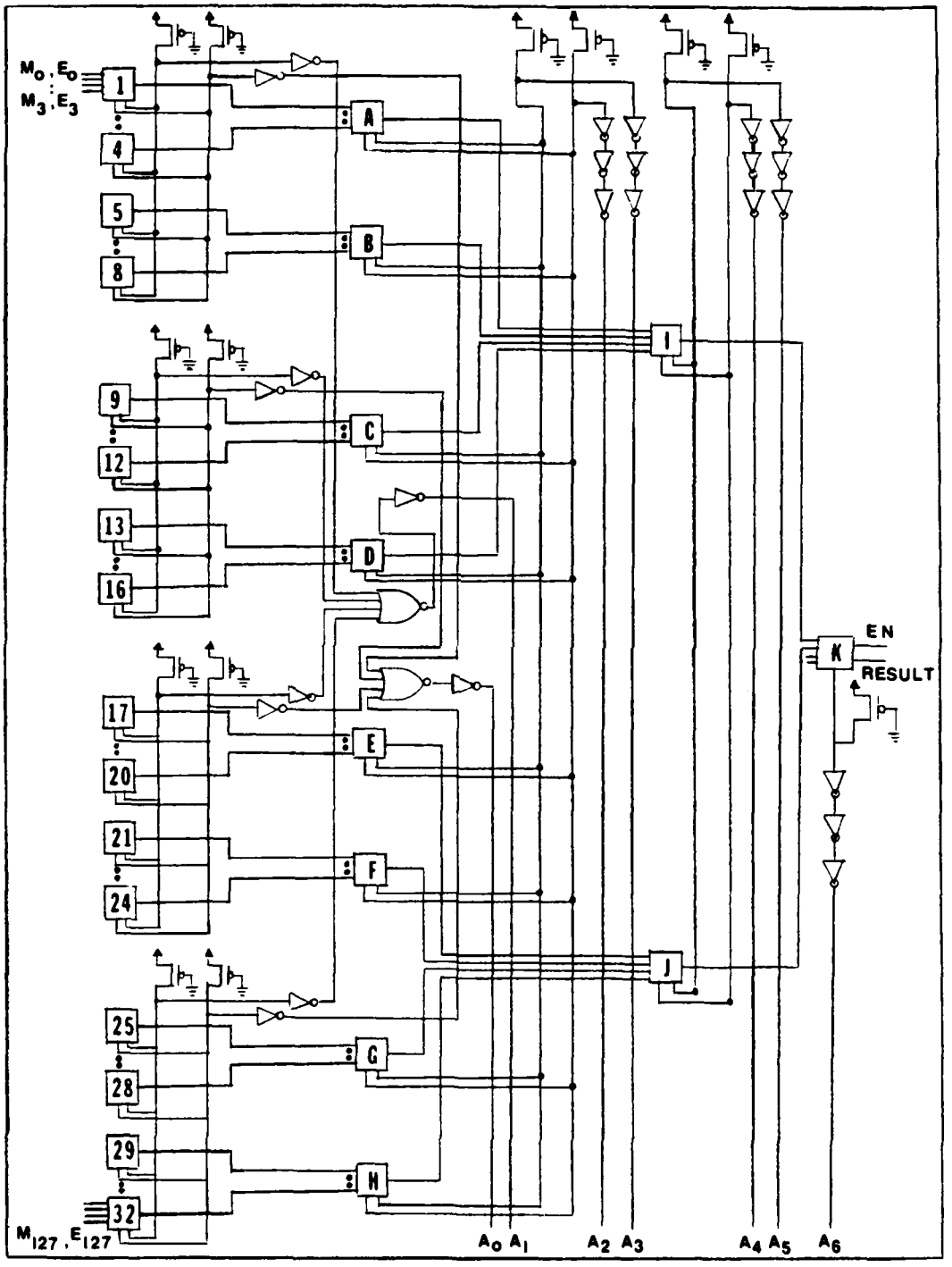


Figure 3.19 Multiple Match Resolver

design as shown in Figure 3.19. The logic diagram of each module was shown in Figure 2.11 of Chapter 2. There are four levels in the tree structure for the MMR. In the first level there are 32 modules,each module taking 4 inputs of the AFIT CAM's 128 words. There are eight modules in second level, two in third and only one module is in the last level. Only half of this last module, K, is used. In the last module, K, master enable signal (EN) and match result signal (RESULT) are connected to the external input and output pads. The input control signal EN enables or disables the MMR. The output signal RESULT will be zero if there are no matches at all.

The match address can easily be derived using wired-OR logic as shown in Figure 3.19. Because of the heavy load on the first level, the generation of outputs A_0 and A_1 are different from the rest. The first level is divided into four groups and each group consists of eight modules. Each group generates A_0 and A_1. These signals are fed into the four input NOR gates and the final addresses A_0 and A_1 are obtained by inverting the output signals of the NOR gates.

3.5.6. One Module of MMR

There are five inputs and seven outputs from one module of the MMR as shown in Figure 2.11. The inputs are the four match signals M_0, M_1, M_2 and M_3, and the enable signal Ea. The outputs are E_0, E_1, E_2, E_3, two address signals A_M and A_L, and R. The logic equations for all the outputs in terms

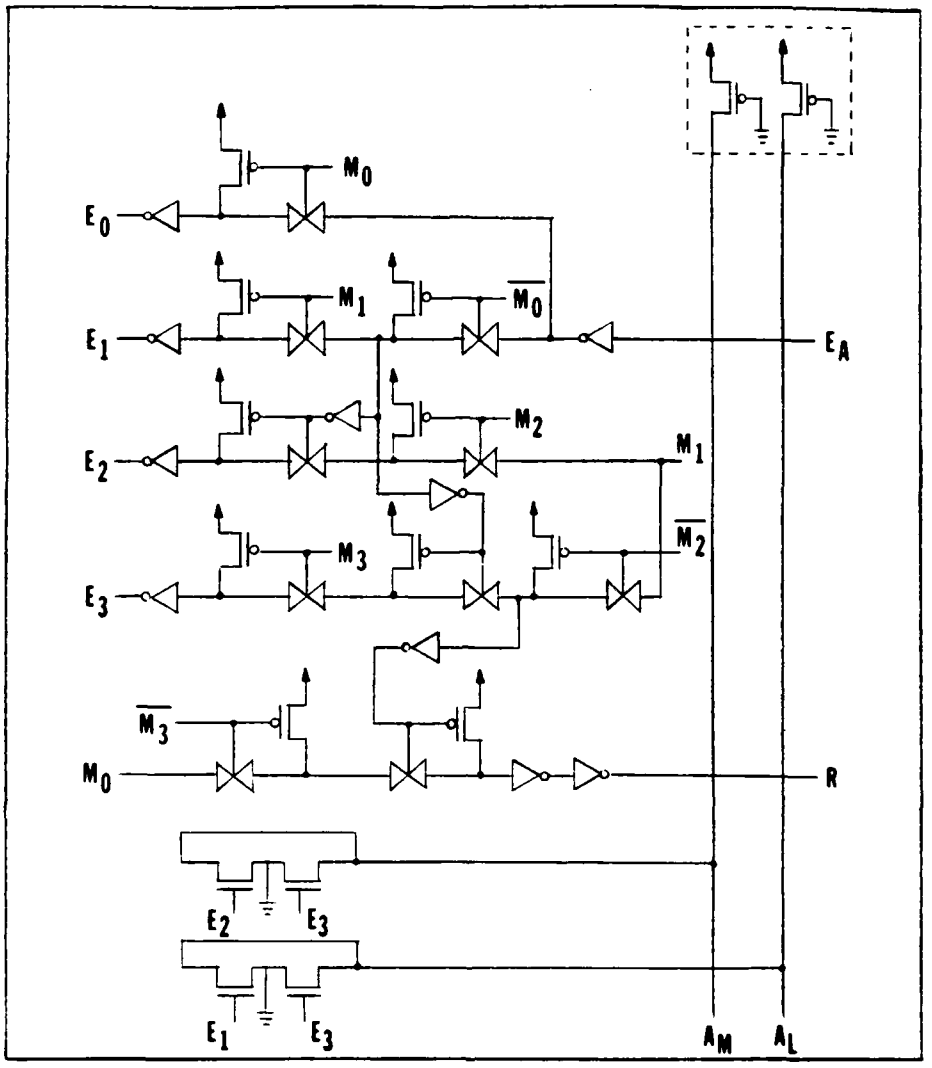


Figure 3.20 One Module of MMR

of the inputs are:

$$E_0 = M_0\ E_A = \overline{\overline{M_0} + \overline{E}_A}$$

$$E_1 = \overline{M}_0\ M_1\ E_A = \overline{\overline{M}_1 + (M_0 + \overline{E}_A)}$$

$$E_2 = \overline{M}_0\ \overline{M}_1\ M_2\ E_A = \overline{(M_1 + \overline{M}_2) + (M_0 + \overline{E}_A)}$$

$$E_3 = \overline{M}_0\ \overline{M}_1\ \overline{M}_2\ M_3\ E_A = \overline{\overline{M}_3 + (M_1 + M_2) + (M_0 + \overline{E}_A)}$$

$$R = M_0 + M_1 + M_2 + M_3 = \overline{\overline{(M_1 + M_2) + (M_0 + M_3)}}$$

$$A_M = E_2 + E_3$$

$$A_L = E_1 + E_3 .$$

We note that $M_0 + \overline{E}_A$ is common to three equations and $M_1 + M_2$ is commmon to two. This way we can simplify the complexity of the overall circuit. The inverted logic is useful because the inverter can make weak signals strong. The logic circuit diagram is shown in Figure 3.20. The pull-up devices in the dotted box are separate from this module. The address lines A_M and A_L can be wired-OR for all modules in each level of the tree.

3.6. Floor Plan

The floor plan of AFIT CAM is shown in Figure 3.21. The organization of the CAM Array is 128 words wide by 16 bits in length. The argument, data, and mask Registers are situated above the CAM Array. The Sense Amplifier and Control Logic for Writing are above the CAM array as shown. The p-devices, for precharging both bitlines for reading are just below the CAM array. The Wordline Driver is placed in with the Address Decoder. The Auxiliary Logic and the Temporary Match Store are part of the Match Store. The MMR has two columns of modules. The first column is the first level of the MMR tree structure as was shown in Figure 3.19. The next column includes the rest of the modules and the pull-up devices. The 2:4 decoder for F_0 and F_1 is placed in between the modules of the second column of the MMR.

Tri-state I/O pads are used for address and data busses.

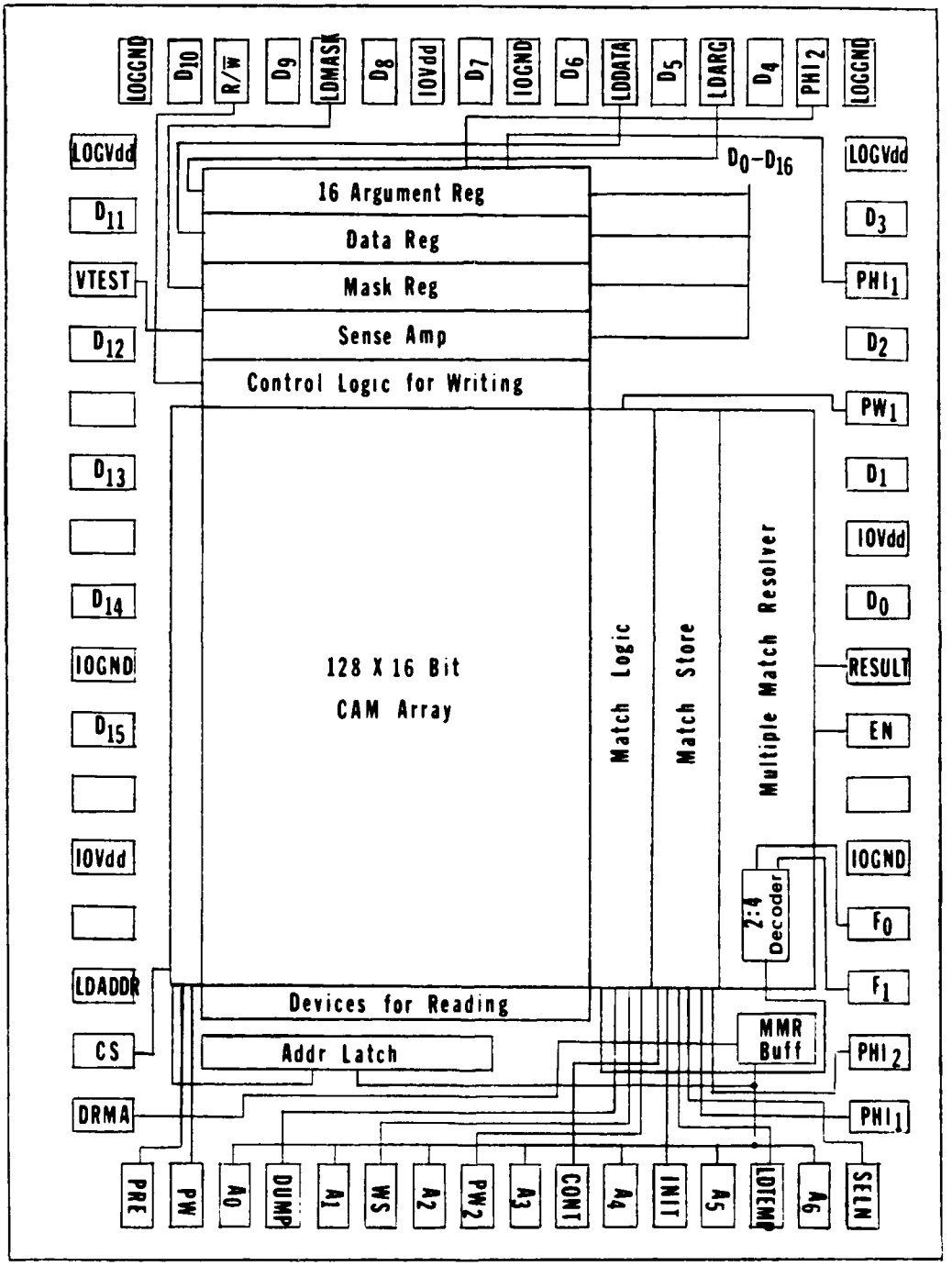


Figure 3.21 Floor Plan of AFIT CAM

There is only one other output signal from the chip, the RESULT signal out of the MMR. There are two separate clock inputs to the chip as well as other redundant pins for V_{dd} and GND signals. Only five unused pads remain in this layout, with the rest of the pads used for input control signals.

3.7. Layout and fabrication of AFIT CAM

SPICE simulations were used extensively, prior to the layout of the circuit to model each cell and its supporting hardware, to observe the performance of the circuitry in an operating environment (e.g., all loading of peripheral circuits).

SPICE was very helpful during the design of the RAM circuitry. SPICE was extensively used to design the sense amplifier and helped determine the size of the precharging devices for writing and pull-down and pull-up devices for reading. A complete transient analysis of the read and write functions was performed by SPICE.

The sizes of pull-down and pull-up devices were given in Section 3.4.4. The channel length and width of the precharging devices (P4, P4, P5 in Figure 3.5) are 3u * 42u.

The time for precharging both bitlines takes 22 ns from the SPICE simulation. And the simulation time for sensing the stored data in the memory cell after precharging is about 10 ns maximum.

The time to change a specific word's value in memory

after either bitlines is completely pulled-up or pulled-down was estimated from simulation to be less than 5 ns. The worst case time to charge 64 words in parallel was determined from simulation to be 60 ns. The Magnitude Detector and the Equality Indicator were simulated to calculate propagation delay times of 160ns and 28 ns respectively.

The circuit was laid out using CAESAR based on CMOS scalable design rules. CAESAR is an interactive system for editing VLSI circuits at each level of the mask geometries.

Layout design rules were checked by LYRA. Cells were checked upon during the design process and upon completion. It was not possible to run the whole layout of AFIT CAM through LYRA because of its size.

MEXTRA was used as a conductivity checker by using its file.al and file.log files.

Table 3.2 Characteristics of the AFIT CAM

Memory Structure	128w x 16b
Supply Voltage	5V
Number of devices	78,000
Die Size	7.9 x 9.2 mm
Process Technology	3u CMOS, 2-layer metals
Number of pins	64

3.8. Characteristics and the pinouts

The major charateristics of the AFIT CAM are shown in Table 3.2. The CAM cell size is 58.5 x 268.5 um with overall chip dimensions of 7.9 x 9.2 mm, which fits into a standard 64-pin DIP. The number of devices in the chip is about 78,000. This is the largest chip ever designed at AFIT, and

also the biggest chip that MOSIS can currently support. The 3u CMOS process technology is used with 2-layer metals.

The pin assignment for AFIT CAM is shown in Figure 3.22, and a photomicrograph of the chip is shown in Figure 3.23.

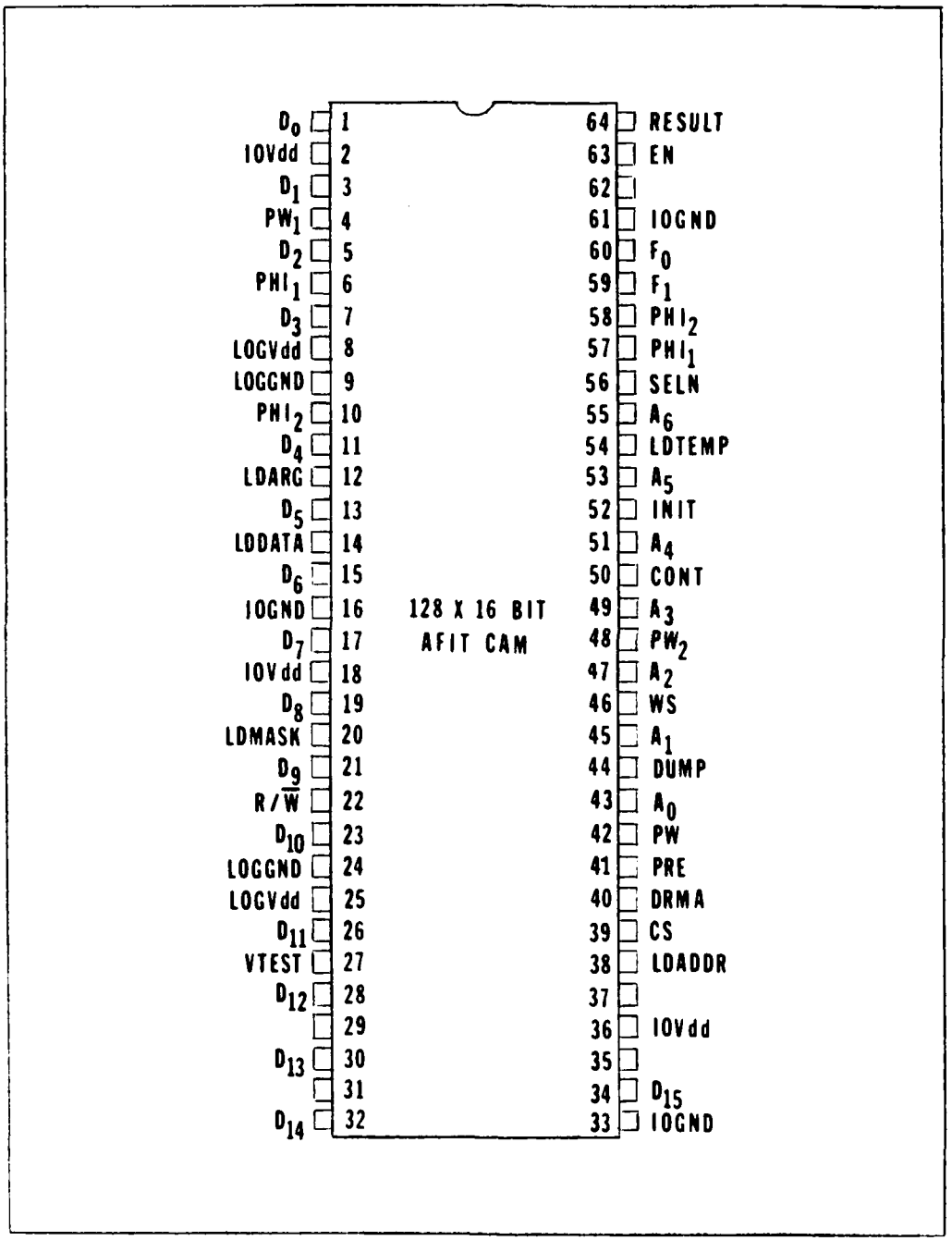


Figure 3.22 Pin Assignments

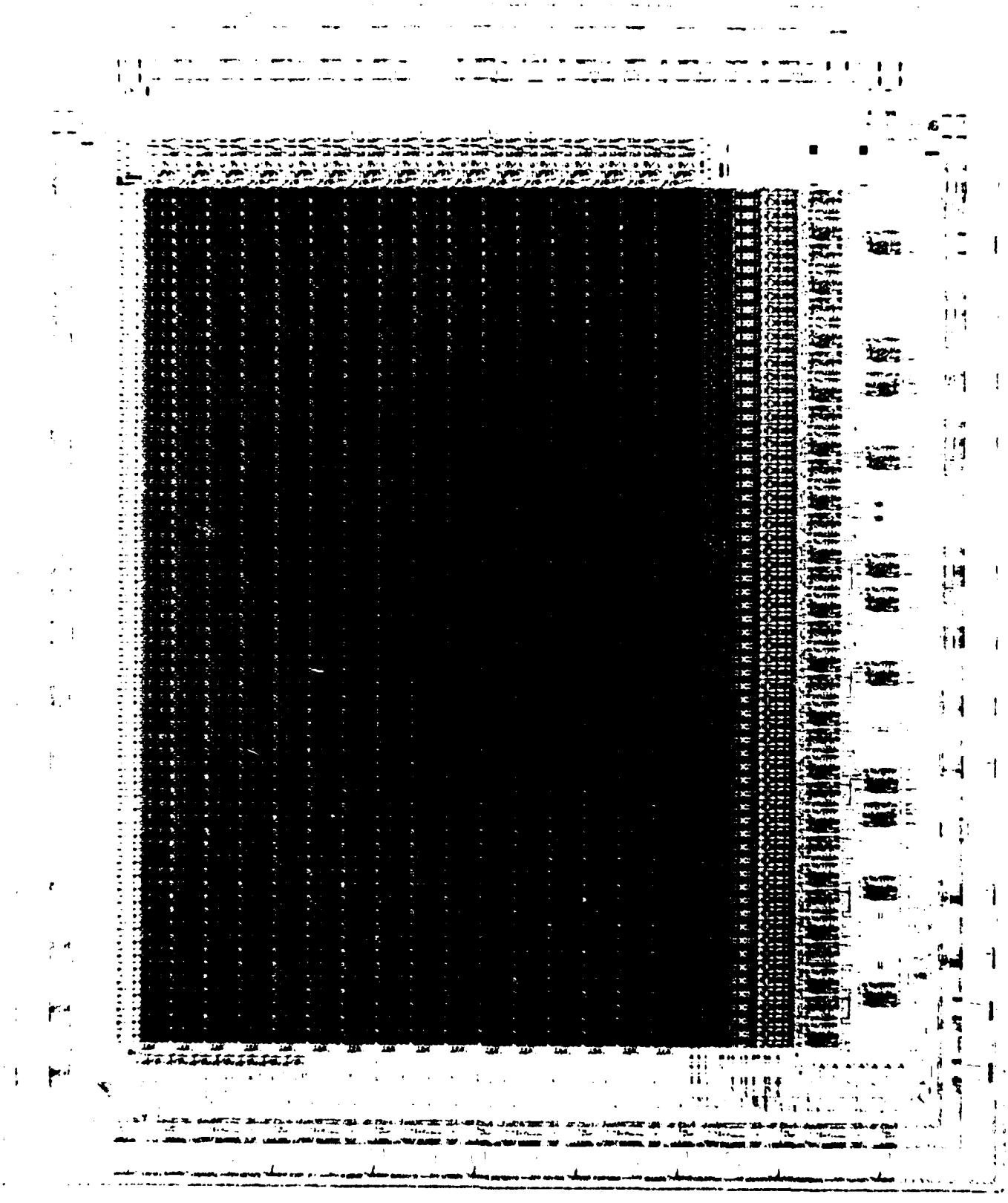


Figure 3.23 Photomicrograh of AFIT CAM

4. Operations of the AFIT CAM

We dicussed the actual hardware implementation of the AFIT CAM and examined the design of each circuit in Chapter 3. In this chapter we will discuss how the AFIT CAM operates for reading, writing and searching the data. Also a DUMP operation will be discussed.

4.1 Read operation

The timing diagram of the read cycle is shown in Figure 4.1. Notice the read-cyle time, T_{CYR}. This represents the minimum amount of time required to read data out of CAM. The precharging time of the bitlines are denoted by T_{PRE}. T_{OD} is the time required to have valid data in the output of the sense amplifier after the wordline is selected.

Prior to starting a read cycle V_{test}, CS, and PW must be set acording to the timing diagram. PW must be set to zero.

The V_{test} should be set to around 2.5V. This is an optimum operating voltage for the sense amplifier to have stable and maximum voltage gain according to the SPICE simulation. V_{test} controls the current source of the sense amplifier and it is connected to the gates of the current source devices Q_5 and Q_6 as shown in Figure 3.10. The sense amplifier is very sensitive to the gate signals of Q_5 and Q_6 so V_{test} should be adjusted by the user to establish the actual optimum voltage.

The CS must be zero before a read cycle starts. The wordline must not be selected until both bitlines are

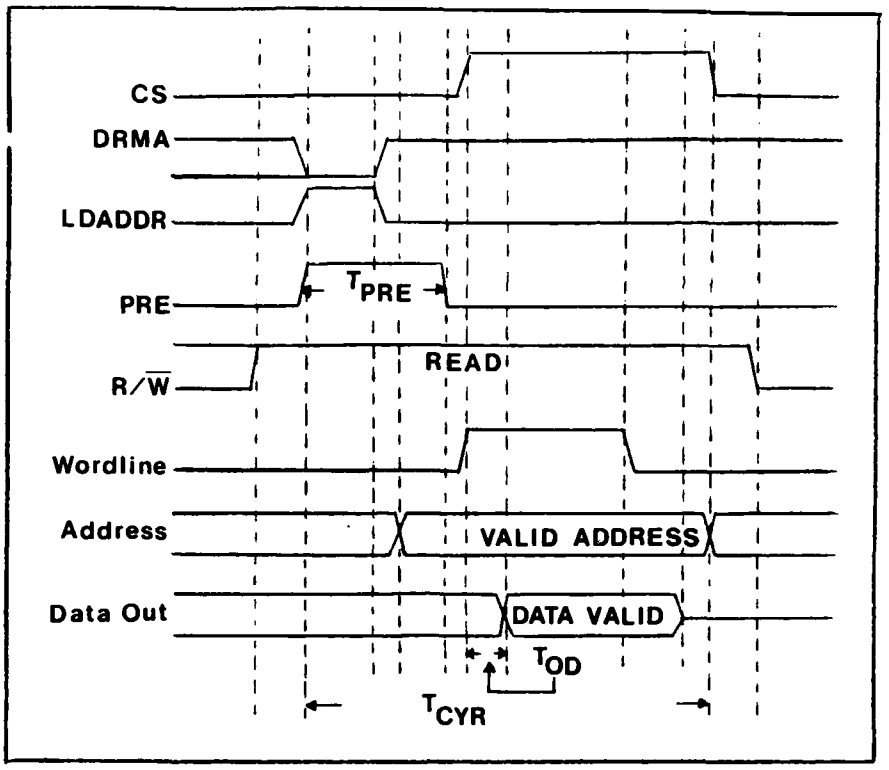


Figure 4.1 Read-Cycle Timing

precharged to V_{dd}. The output of the sense amplifier is gated on when CS = 1.

PW must be zero during the read-cycle. This allows the transmission gate in the wordline driver to turn on.

After the initial conditions are set the bitlines are precharged to V_{dd}. This is done by opening the precharging p-devices P3, P4, and P5 in Figure 3.5. To do this, both PRE and R/W must be one as shown in Figure 4.1

At the same time, the DRMA becomes zero to place the address inputs on the address bus from the address pads (A_0 - A_6). Control signal, LDADDR latches the address in the Address Latch and drives the address decoder. The valid

address is available at this point.

Precharging can be terminated by setting the PRE signal to zero which causes the Wordline Driver and Wordline to be selected. As soon as pass transistors in the memory cell are opened by the wordline the sense amplifier senses the differential voltage on the both bitlines. The valid data output is then available on the data bus.

CS must stay high until all the valid data is read and driven out to the data pads (D_0 - D_{15}). CS controls the tri-state I/O pads, and data can be driven out of the chip only when CS is one.

Read-cycle operation discussed so far uses an external address. However, the address can be obtained internally from the MMR Address Buffer. To unload the MMR Buffer, the control signal to the MMR Buffer, Drive Match Address (DRMA) must be one.

Note that during the read cycle the pull-up and pull-down devices P1, P2, N1, and N2 are off. These are used only for writing into the CAM.

4.2. Write Operations

The AFIT CAM can handle two kinds of write operations. One is to write the CAM like an ordinary RAM. The other one is parallel writing to multiple words. This feature was described in Section 4.1.

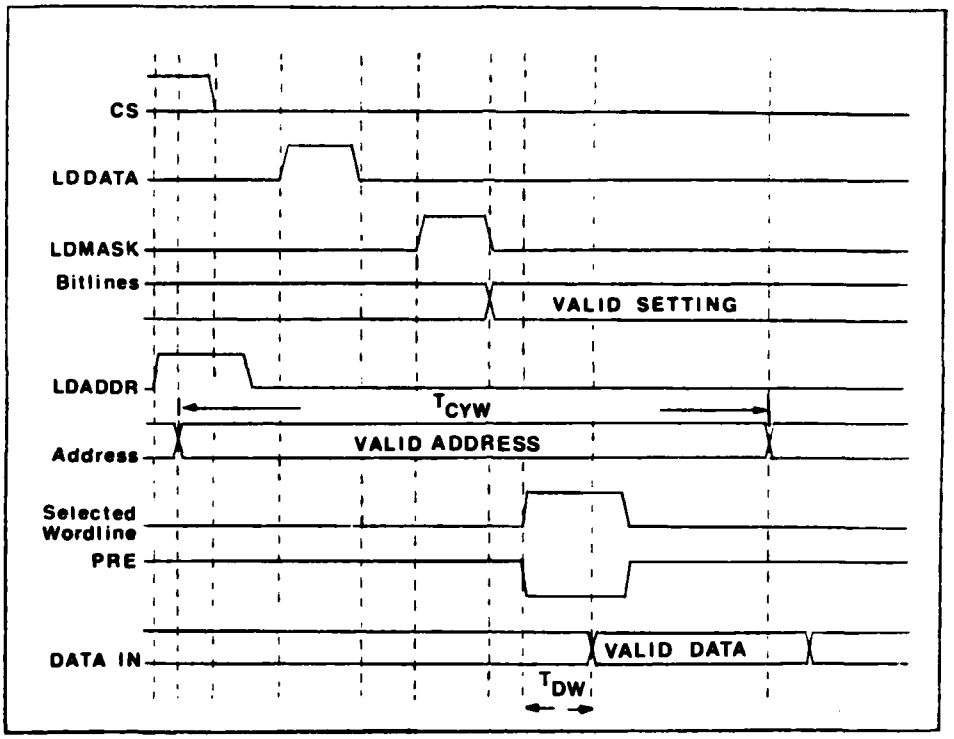


Figure 4.2 Write-Cycle Timing

4.2.1 Write one word

The timing for the write cycle is shown in Figure 4.2. The T_{CYW} represents the minimum amount of time that must be allocated for writing data into the CAM. Write-cycle time is simply the sum of the switching delays of the internal circuitry involved in the write operation.

Assume that the address is on the address bus. The control signal of Address Latch, LDADDR, latches the address and drives it to the decoder. At this point, the valid address is in the Wordline Driver. We cannot drive the wordline until both bitlines have correct settings.

The Control Logic for Writing circuitry controls the

pull-up or pull-down devices. This circuit takes its inputs from the data and mask register. The data register stores the data to be written and the mask register holds masking information. The operation of Control Logic for Writing circuitry is discussed in detail in Section 3.4.4.

CS must be "0" to place the data (D_0 - D_{15}) on the data bus, and it must be remain "0" until the end of write cycle. A low value of CS allows the wordline to be selected.

Both bitlines will have valid settings as soon as the mask register is loaded as shown in Figure 4.2. Soon after both bitlines are precharged the PRE signal triggers the wordline to be selected. The data is written after opening up the pass transistors in the memory cell. This time is represented by T_{DW}.

4.2.2. Parallel Write

The AFIT CAM will write to all words in the memory that have a one in the Match Store during a Parallel Write. Of course R/W must be zero for this operation. The Parallel Writing is accomplished in two steps - 64 words at a time.

In first step, the words in the upper 64 words of the CAM array which have the one in the Match Store are written. The write cycle is same as the write operation discussed in previous section except the wordline is selected by PW1 control signal and driven from the Auxiliary Logic shown in Figure 3.17.

In next step, everything is same as step one except PW2

must be "1" instead of PW1. Note that both PW1 and PW2 must *not be one during* this operation because the pull-up and pull-down devices (P1, P2, N1, N2 in Figure 3.5) are not big enough to drive all 128 word simultaneosly.

4.3 Search Operations

We will examine the two kinds of search operations in this section, "simple search" and "complex search". The simple search is defined as a search that the AFIT CAM supports directly on the hardware. These match operations are EQ, NE, GE, and LT. The complex search is defined as a search that the AFIT CAM requires more than one cycles per search operation.

4.3.1. Simple Search

The chart showing the sequences and set-ups for the simple search is shown in Figure 4.3. First, initialize the Match Store to "1". The MMR functions only when master enable signal, EN, is "1". The Word Select (WS) signal must "0", otherwise the Auxiliary Logic does not forward the match signal to the Temporary Match Store.

Next, load the argument and mask registers. Then select one of the comparison types: EQ (F_0F_1 = 00), NE (F_0F_1 = 01), GE (F_0F_1 = 10) and LT (F_0F_1 = 11).

The Match Logic will then output the match signal. These match responses will be stored in the Temporary Match Store.

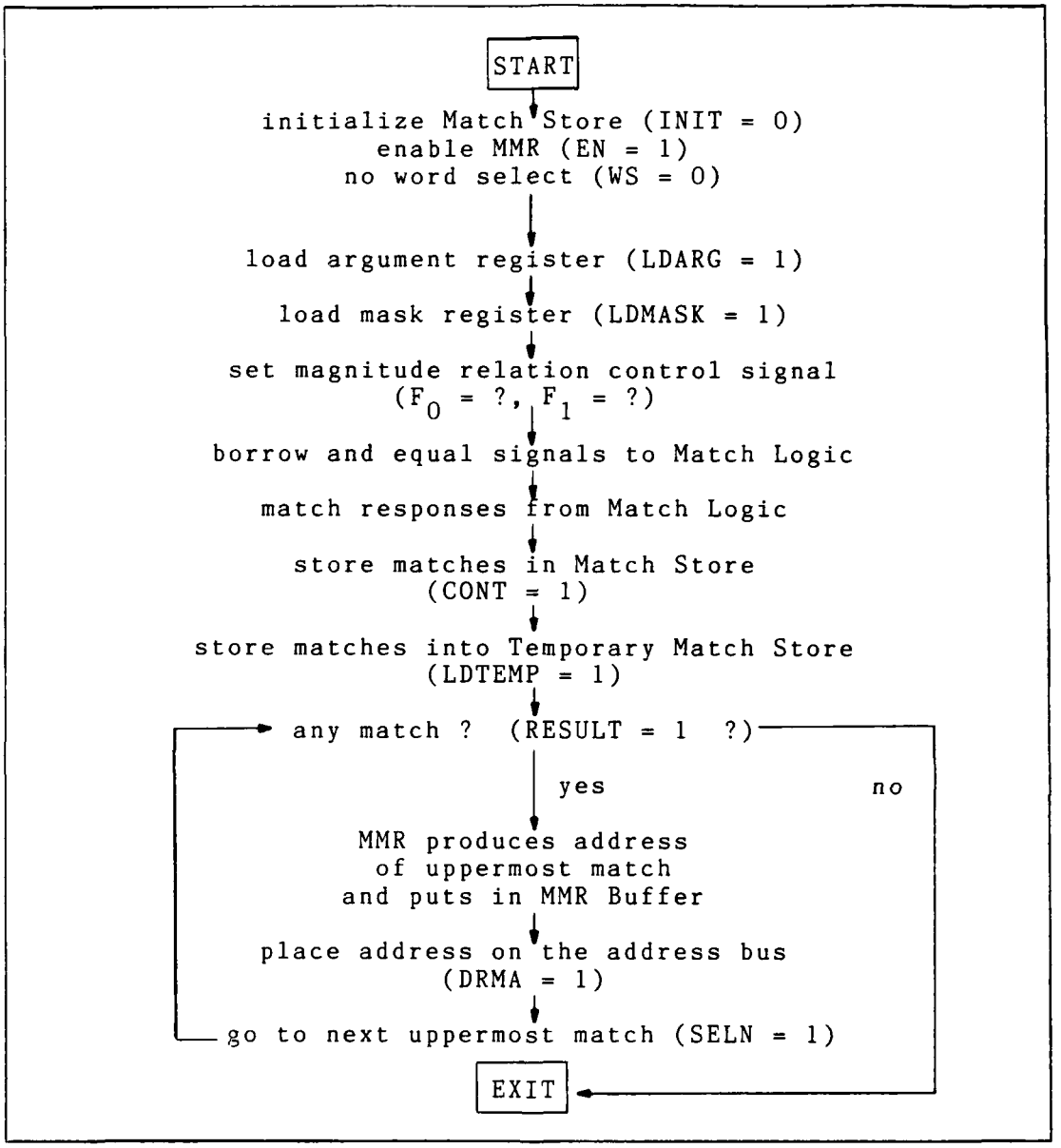


Figure 4.3 Simple Search

The uppermost addresses will appear at the bottom of the MMR in the sequence of top down order. Figure 4.3 summarizes the simple search algorithm using the AFIT CAM.

4.3.2. Complex Search

Figure 4.4 shows the complex search algorithm. Searching for the numbers in the memory which are greater-than (GT) or less-than-equal (LE) is not a simple search. These particular match operations are not directly implemented in the AFIT CAM chip. This types of search can, however, be done on the AFIT CAM by the algorithm shown in Figure 4.4.

Between-limits searchs (e.g., X > A >= Y) can be done if we take two more search cycles (A is a set of stored numbers in the memory cell and X, Y are argument lists).

First, we perform the simple search, A < X as described in the privious section. Second, we do the other simple search, A >= Y. The resulting match responses are ANDed and stored in the Match Store and the Temporary Match Store. This ANDing operation allows us to perform much more complicated searching than the example shown above.

Note that Continue-to-Match (CONT) control signal in the Match Store can be left open until several ANDing operations have been done.

4.3.3. DUMP operation

This operation resets all the 16th bits in the stored data words to zero if the word has the corresponding Match Store set to "1". This operation takes precedence over any write or read. To prevent fighting inside the memory cells, write or read operations must not taken place at the same time. When the DUMP signal is "1" and match in the Match

Store is "1" then the flip-flop in the last memory cell is simply pulled-down to ground as shown in Figure 3.17.

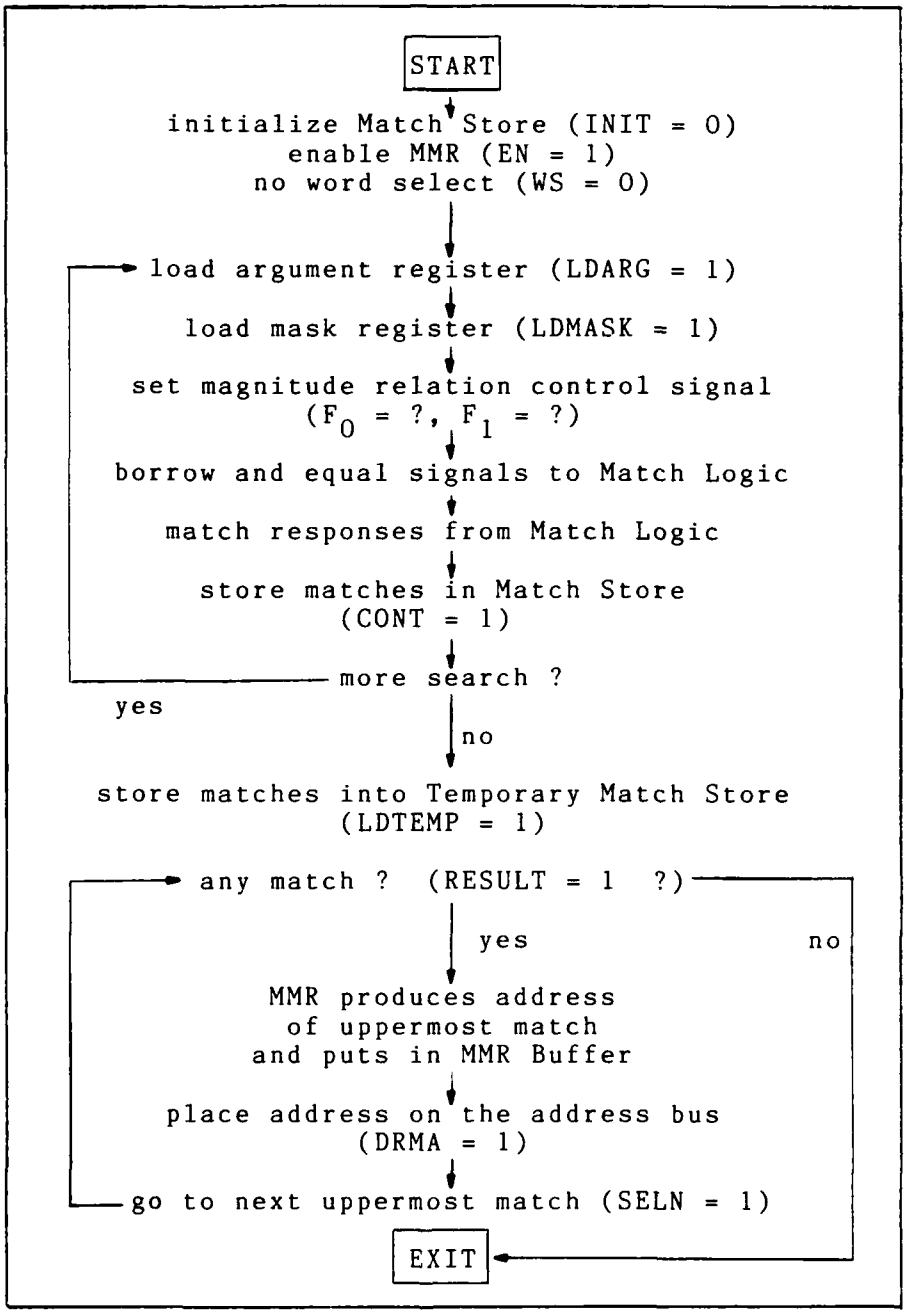


Figure 4.4 Complex Search

5. Results, Conclusion and Recommendations

5.1. Results

The all-parallel AFIT CAM has been designed and fabricated with high performance, excellent device density, and unique features which allow it to support important applications in artificial intelligence, image recognition, and general computer architecture. The AFIT CAM has total of 78,000 devices and achieves a density of 1,073 devices per mm^2. It has 2K memory structure and fits in 64 pin DIP.

In this section, we will discuss the characteristics and performanc of the CAM cell, the RAM circuitry, and the CAM circuitry. The performance measures given here are based on the SPICE simulations and remain to be verified by testing.

5.1.1. High Density CAM Cell

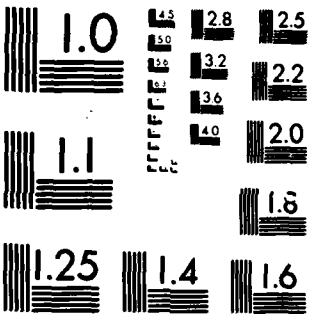
The CAM cell consists of a memory cell and comparison logic circuitry which has a magnitude detector and an equality indicator. The size of the CAM cell is 58.5 um X 268.5 um and a density of 1655 devices per mm^2.

The memory cell uses a 6-transistor CMOS static RAM cell. This cell features low power dissipation, correct operation over a wide range of supply voltages and temperatures, excellent noise immunity.

The magnitude detector which generates the Borrow signal employs the CMOS inveror transmmission gate logic (CITGL). Thus, it has a low power consumption and high switching speed. The propagation delay of Borrow signal through 16

AD-A163 995 DESIGN AND IMPLEMENTATION OF HIGH PERFORMANCE CONTENT-ADDRESSABLE MEMORIES(U) AIR FORCE INST OF TECH WRIGHT-PATTERSON AFB OH SCHOOL OF ENGINEERING 2/2

UNCLASSIFIED W H SHINN DEC 85 AFIT/GE/ENG/85D-39 F/G 9/5 NL



CAMs cell is only 160 ns.

5.1.2. RAM Circuitry

The performance of the RAM circuitry in the AFIT CAM is comparable to that of commercially available state-of-the-art CMOS RAMs. The worst case address access time is 22 ns. The times for reading and writing take only 22 ns and 20 ns respectively. The worst case parallel writing time is 60 ns (when writing to 64 address locations simultaneously).

The AFIT CAM has unique writing features. They are maskable and parallel writings. The bits of a words can be written selectively according to the information in the mask register. Also words can be written to one CAM operation.

The Address Latch and MMR Buffer allow address to be obtained externally or internally. The tri-state I/O address pads reduce the number of pins and allow the match address to be sent off-chip.

special care was taken to design and layout the RAM circuitry, especially sense amplifier because its its analog behavior. The size of the precharging devices and the pull-up and pull-down devices was calculated after running a series of SPICE simulations under several operating conditions.

The density of this circuitry is not as high as other circuitry in the CAM because of large driver transistors are required.

5.1.3. CAM Circuitry

This circuitry was designed to perform magnitude comparison and to generate the match address on the chip.

The CAM chip supports four types of comparisons directly on the chip. They are equal, not-equal, greater-than-or-equal, and less-than. In addition, it can reset the last memory cell in the word locations which mathched. This DUMP operation allows garbage collection in LISP to aid artificial intelligence applications.

The tree structure of Multiple Match Resolver (MMR) circuitry replaces the priority network and the address generator and encoder. A significant savings in chip area is realized by using this circuitry. Most importantly it allows the uppermost match to be found in $O(\log_4 N)$ time rather than $O(N)$ time where N is the number of words to be resolved (128 for the 3 mm CAM).

The propagation delays of magnitude detector and equality indicator are 160 ns and 27 ns respectively. This allows it to compute 800 million magnitude comparisons per second or 3.2 billion equality comparisons per second. The difference is significant since only equality comparison is need for many applications.

5.1.4. Chip Fabrication Results

The AFIT CAM has been fabricated by MOSIS and tested in the VLSI Laboratory at AFIT.

During the power on test, we found that the current-

voltage between Logic V_{dd} and Logic GND power supply characteristics show that the chip had a short. To locate the short, we examined the entire chip with a microscope. In addition, CAESAR design files were reviewed and the mextra circuit was rerun. This work gave no indication of a short circuit inside the chip. Howerver, we detected some unrelated minor design errors.

During microscopic examination, we found that the second metal lines were much closer than expected. In actual layout, we used 3u width and 6u separation for second metal. However, we ran second metal expander program just before submission of the cif file to MOSIS. This set both width and separation of second metal to 4.5 um. However, the width and the separation appears to be 7 um and 2 um on the fabricated chip. The second metal must have been bloated by the manufacturer. The MOSIS 3 um design rule for second metal is 5 um width and 5 um separation. Eventhough we reduced this by 0.5 um, the remaining separation was still too small.

At this point, we suspected that over bloated second metal lines caused the short. The minor errors found during testing have been corrected and the cif file has been resubmited to MOSIS for fabrication.

5.2. Conclusions

This thesis effort produced the design and implementation of a high performance all parallel content-

addressable memory. Efficient utilization of silicon area enables all of the circuitry in the AFIT CAM to be implemented on a sigle chip using a 3 micron CMOS process. This chip can search a maximum of 3.2 billion words per second and achieves a overall device density of 1,073 devices per mm^2 which is very high for VLSI circuits.

Power consumption was kept to a minimum by employimg the CMOS technology and making maximum use of static inverter transmmission gate logic throughout the design.

By utilizing its unique features (DUMP, Parallel Write, Maskable Write), it supports applications in artificial intelligence and image processing. In general the AFIT CAM can be used in areas which require extensive searching.

5.3. Recommendation

Testing the second pass of the AFIT CAM is required to determine correct chip functionality. Also, determination of the available input voltage range of the V_{test} control signal to sense amplifier is required.

The integral part of the AFIT CAM is the CAM cell and this takes well over half of the entire chip area. If we reduce the size of CAM cell, we could save chip area which can be utilized for other useful circuits.

A second version of the CAM cell has been redesigned and layed out. This new cell is 25% smaller than the present one. It realized the XOR gate with two transistors and has only one inverter for transfering the Borrow Out signal to

next stage. The new CAM cell will have more power dissipation and weak ones and zeros. SPICE simulations must be done before speed comparison can be done.

In the next generation of the AFIT CAM, two more magnitude comparison logic circuits should be added on the chip (greater-than and less-than-or-equal).

Alos a circuit to count the number of match responses should be developed. One circuit consider is a analog circuit which provide a vertical line with 128 pull-down devices gated by the match signals. The sensing of sunk current can be done external to the chip to determine the number of responses.

BIBLIOGRAPHY

1. Kohonen, Teuvo. Content-Addressable Memories. Berlin: Springer-Verlag Publishers, 1980.

2. Hwang, Kai. Computer Architecture. New York: McGraw-Hill Book Company, 1984.

3. Foster, Caxton C. Content Addressable Parallel Porccessors. New York: Van Nostrand Reinhold Co., 1969.

4. Lewis, E.T. CMOS Custom Circuit Design. Lecture material distributed in EE6.98, VLSI Design. AFIT, Wright-Patterson AFB OH, December 1984.

5. Grebene, Alan B. Bipolar and MOS Analog Integrated Circuit Design. New York: John-Wiley & Sons, 1984.

6. Mano, Morris M. Computer System Architecture. New Jersy: Prentice-Hall, 1982.

7. Mavor, J. and Jack, M. A. Intoroduction to MOS LSI Design. Springer-Verlag: New York, 1980.

8. Mead, Carver and Conway Lynn. Introduction to VLSI Systems. Massachusetts: Addison-Wesley, 1980.

9. Carr, William N. and Mize,Jack P. MOS/LSI Design and Application. New York: McGraw-Hill, 1972.

10. Winston, Patrick H. Artificial Intelligence. Massachusetts: Addison-Weisley Publishing Co., 1984.

11. Barr, Avron and Feigenbaum, Edward A. The Handbook of Artificial Intelligence. Los Altos: William Kaufmann Inc., 1981.

12. Higgins, Richard J. Electronics with Digital and Analog Integrated Circuits. New York: Prentice-Hall, 1983.

13. Hamacher, Carl V. and others. Computer Organization. New York: McGraw-Hill Book Company, 1978.

14. Newkirk, John and Mathews, Mathews. The VLSI Designer's Library. Massachusetts: Addison-Wesley Publishing Co., 1983.

15. Weste, Neil and Kamram. Principle of CMOS VLSI Design: A System Perspective. Massachusetts: Addison-Wesley Publishing Co., 1984.

16. Einspruch, Norman G. VLSI Electronics. Microstructure Science, Vol 5. New York: Academic Press, 1982.

17. Minato, Osamu and others. "A 20 ns 64K CMOS Static RAM", IEEE Journal of Solid-State Circuit, sc-19, No 6: 1008-1013 (December 1984).

18. Yamamoto, Sho. "High Density SRAMs", 1985 IEEE International Solid-State Circuit Conference. 57-69 (February 1985).

19. Kodata, Hiroshi. "An 8Kb Content-Addressable and Reentrant Memory", 1985 IEEE International Solid-State Circuit Conference. 42-43 (February 1985).

20. Rowe, Mark C. "Content-Addressable Memory Manager Desigen and Evaluation". Master Thesis. AFIT Fall 1985.

21. Soclof, Sidney. Analog Integrated Circuits. New Jersey: Printice-Hall, 1985

22. Anderson, George A. "Multiple Match Resolver: New Design Method", IEEE Trans. on Computers, c23, no.12: 1317-1320 (December 1974)

23. Crane, Bently. "Path Finding with Associative Memory." IEEE Trans. on Computers, c17, no.7, 691-693, (July 1968)

VITA

Wook Hyun Shinn was born in Taegu, Republic of Korea on September 28, 1952. A year after graduation from Kae-Sung High School, he immigrated to the United States of America in 1972. He enlisted in the USAF in 1976 and was selected to a Airman Education Comminssioning Program in 1979. He attended the University of Arizona where he received the degree of Bachelor of Science in Electrical Engineering in 1981. After graduation, he was commissioned a second lieutenant in the USAF through the Officer Training School. From October 1981 through May 1984, he stationed in Kirtland AFB, New Mexico where he served as an electical engineer in the Air Force Weapons Laboratory. In June 1982, he entered the School of Engineering, Air Force Institute of Technology to pursue a graduate degree in electrical engineering specializing in semiconductor technology and VLSI design.

Permanent Address: 577 27th Street
San Francisco, California 94121

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE

REPORT DOCUMENTATION PAGE

1a. REPORT SECURITY CLASSIFICATION UNCLASSIFIED	1b. RESTRICTIVE MARKINGS
2a. SECURITY CLASSIFICATION AUTHORITY	3. DISTRIBUTION/AVAILABILITY OF REPORT Approved for Public Release; Distribution Unlimited
2b. DECLASSIFICATION/DOWNGRADING SCHEDULE	
4. PERFORMING ORGANIZATION REPORT NUMBER(S) AFIT-GE/ENG/85D-39	5. MONITORING ORGANIZATION REPORT NUMBER(S)
6a. NAME OF PERFORMING ORGANIZATION School of Engineering 6b. OFFICE SYMBOL (If applicable) AFIT/ENG	7a. NAME OF MONITORING ORGANIZATION
6c. ADDRESS (City, State and ZIP Code) Air Force Institute of Technology Wright-Patterson Air Force Base, OH 45433	7b. ADDRESS (City, State and ZIP Code)
8a. NAME OF FUNDING/SPONSORING ORGANIZATION AFOSR 8b. OFFICE SYMBOL (If applicable)	9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER
8c. ADDRESS (City, State and ZIP Code) Bolling Air Force Base, Washington, DC 20332	10. SOURCE OF FUNDING NOS. PROGRAM ELEMENT NO. / PROJECT NO. / TASK NO. / WORK UNIT NO.
11. TITLE (Include Security Classification) See Block 19	
12. PERSONAL AUTHOR(S) Wook H. Shinn, B.S., Capt, USAF	
13a. TYPE OF REPORT MS Thesis 13b. TIME COVERED FROM _____ TO _____	14. DATE OF REPORT (Yr., Mo., Day) 1985 December 15. PAGE COUNT 101
16. SUPPLEMENTARY NOTATION	

17 COSATI CODES		
FIELD	GROUP	SUB. GR
09	01	

18. SUBJECT TERMS (Continue on reverse if necessary and identify by block number)
Content-Addressable Memories, CAM, Associative Memories, VLSI, CMOS, Searching, Artificial Intelligence

19. ABSTRACT (Continue on reverse if necessary and identify by block number)

TITLE: DESIGN AND IMPLEMENTATION OF HIGH PERFORMANCE CONTENT-ADDRESSABLE MEMORIES

THESIS ADVISOR: RICHARD W. LINDERMAN, CAPT, USAF

Approved for public release: IAW AFR 190-1.
LYNN E. WOLAVER 16 JAN 86
Dean for Research and Professional Development
Air Force Institute of Technology (AFIT)
Wright-Patterson AFB OH 45433

20 DISTRIBUTION/AVAILABILITY OF ABSTRACT UNCLASSIFIED/UNLIMITED ☒ SAME AS RPT ☐ DTIC USERS ☐	21 ABSTRACT SECURITY CLASSIFICATION UNCLASSIFIED
22a. NAME OF RESPONSIBLE INDIVIDUAL Richard W. Linderman, Capt, USAF	22b TELEPHONE NUMBER (Include Area Code) 513-255-6913 22c OFFICE SYMBOL AFIT/ENG

DD FORM 1473, 83 APR EDITION OF 1 JAN 73 IS OBSOLETE

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE

The content-addressable memories (CAMs) have major applications in developing areas of artificial intelligence and image processing. This work involves researching, designing, and implementing a high performance, high density CAM. The design, fabrication, and test of this all-parallel AFIT CAM is discussed. The AFIT CAM stores 2K bits of information and fits in 64 pin dual-in-line package. It was fabricated using a 3μ CMOS technology with 2-layer metals.

The AFIT CAM supports four types of comparisons directly on the chip. These are equal, not-equal, greater-than-or-equal, and less-than. In addition, it has DUMP and Parallel Write operations to aid artificial intelligence and image processing applications.

The AFIT CAM has a total of 78,000 devices and achieves a density of 1,073 devices per square millimeter. The memory access time for reading and writing are only 22 ns and 20 ns respectively. The CAM chip computes 800 million magnitude comparisons or 3.2 billion equality comparisons per second.

The cif file has been resubmited to the manufacturer for fabrication due to a short circuit which was caused by over bloated second metal lines. A second version of the AFIT CAM cell has been redesigned and nets a 25% reduction in size.

END

FILMED

3-86

DTIC